NoSQL Databases

Sadalage and Fowler
Chapters 1-3

What are NoSQL Databases?

- NoSQL originally meant for databases that were:
  - Non-relational database (not using SQL)
  - Designed to run on cluster servers
  - Open-source
  - Optimized for web use
  - Unrestricted by a formal schema (data structure) definition
- The definition quickly mutated to include databases:
  - With SQL-like query languages (or optional use of SQL)
  - With non-relational schema definitions
  - With marketing staff eager to latch onto the NoSQL moniker
- There is no formal, widely accepted definition

Not Only SQL?

- Many interpret NoSQL to mean “not only” SQL
  - Underscores the notion that NoSQL is not intended to displace relational
  - Has the unfortunate side-effect of including object-relational databases such as Oracle and Postgres
- Authors Pramod J Sandalage and Martin Fowler suggest the term “polyglot persistence”
  - Polyglot refers to different data storage mechanisms
  - Persistence refers to permanent storage of data
  - Organizations use a mix of data storage technologies based on:
    - The nature of the data
    - Requirements for manipulation of the data

The Objectives of Using NoSQL

- Increased application developer productivity using data structures that better match application needs
- More flexibility for persistent data storage compared with relational DBMSs
- Designed specifically to run on clusters, improving scalability (big data) and performance
  - A computer *cluster* consists of a set of loosely or tightly coupled computers that work together, giving the appearance of a single system.
- Potentially more flexible control/resolution of concurrent update issues

Objectives Not Met by NoSQL Databases

Most NoSQL solutions fall short of relational databases in these areas:

- Flexible ad-hoc query capabilities
- Transaction processing and immediate data consistency
- Consolidation of data for multiple applications into a common database
- More robust security features and data access controls
- Data model standardization

Using Both Relational (SQL) and NoSQL

- Relational still the most common for mainstream business transactions (e.g. order management, customer relationship management, payroll)
- NoSQL can be used to augment relational databases for:
  - Massive data volumes (big data)
  - Unstructured data (e.g. social media, web clickstreams, audio/video)
  - Preparing data from display on the web
  - High speed collection of data (e.g. telecommunications circuit utilization)

Categorization of NoSQL Databases

- There are many ways to categorize NoSQL databases.
- One common method is based on the way they organize data. In this topic, we will explore four such categories:
  - Key-value databases
  - Document databases
  - Columnar databases
  - Graph databases

Data Structure: Northwind

Customer File

Customer ID	Company Name	Contact First Name	Contact Last Name	Job Title	City	State
6	Company F	Francisco	Pérez-Olaeta	Purchasing Manager	Milwaukee	WI
26	Company Z	Run	Liu	Accounting Assistant	Miami	FL

Employee File

Employee ID	First Name	Last Name	Title
2	Andrew	Cencini	Vice President, Sales
5	Steven	Thrope	Sales Manager
9	Anne	Hellung-Larsen	Sales Representative

Product File

Product ID	Product Code	Product Name	Category	Quantity Per Unit	List Price
5	NWTO-5	Northwind Traders Olive Oil	Oil	36 boxes	$21.35
7	NWTDFN-7	Northwind Traders Dried Pears	Dried Fruit & Nuts	12 - 1 lb pkgs.	$30.00
40	NWTCM-40	Northwind Traders Crab Meat	Canned Meat	24 - 4 oz tins	$18.40
41	NWTSO-41	Northwind Traders Clam Chowder	Soups	12 - 12 oz cans	$9.65
48	NWTCA-48	Northwind Traders Chocolate	Candy	10 pkgs.	$12.75
51	NWTDFN-51	Northwind Traders Dried Apples	Dried Fruit & Nuts	50 - 300 g pkgs.	$53.00

Order File

Order ID	Customer ID	Employee ID	Order Date	Shipped Date	Shipping Fee
51	26	9	4/5/2006	4/5/2006	$60.00
56	6	2	4/3/2006	4/3/2006	$0.00
79	6	2	6/23/2006	6/23/2006	$0.00

Order Detail File

Order ID	Product ID	Unit Price	Quantity
51	5	$21.35	15
51	41	$9.65	21
51	40	$18.40	2
56	48	$12.75	20
79	7	$30.00	14
79	51	$53.00	8

Key-Value Databases

- Stores data values using a key to identify each record.
  - The value that is stored with each key can be thought of as a blob (binary large object).
  - Hierarchies of data are collapsed into tree structures and formatted using XML, JSON, BSON, or a similar markup or formatting language, yielding a a single data value (a blob).
  - The key-value DBMS knows nothing about the contents of the blob.
    - The application that uses the DBMS must pick apart the blob in order to make sense of it.

Popular Key-Value Databases

- Hadoop Distributed File System (HDFS)
- Riak
- Redis
- Memcached DB
- Berkeley DB
- HamsterDB
- Amazon DynamoDB (not open source)
- Project Voldemort (open source implementation of DynamoDB)

Northwind Tree Structure

Customer ID

Company Name

Employee ID

Order Date

Shipped Date

Shipping Fee

- Product ID
- Product Code
- Product Name
- Unit Price
- Quantity

JSON for Order 51


```
{
"id": 51,
"customerId": 26,
"companyName": "Company Z",
"employeeId" :9,
"orderDate": "2006-04-05",
"shippedDate": "2006-04-05",
"shippingFee": 60.00,
"orderItems": [
  {
  "productId": 5,
  "productCode": "NWTO-5",
  "productName": "Northwind Traders Olive Oil",
  "unitPrice": 21.35,
  "quantity": 15
   },
```

JSON for Order 51 (continued)


```
{
  "productId": 41,
  "productCode": "NWTSO-41",
  "productName": "Northwind Traders Clam Chowder",
  "unitPrice": 9.65,
  "quantity": 21
   },
  {
  "productId": 40,
  "productCode": "NWTCM-40",
  "productName": "Northwind Traders Crab Meat",
  "unitPrice": 18.40,
  "quantity": 2
   }
 ]
}
```

Key-Value Physical Storage

- Once the JSON has been formed by the application, the work that the key-value DBMS must do is quite simple.
- The only operations it needs to support are:
  - Adding a new key-value pair,
  - Updating (overwriting) the value for an existing key,
  - Deleting a key (which also deletes the value).
- Here is how the key-value DBMS would store the data:

KEY	VALUE
51	Order 51 Data (JSON)
56	Order 56 Data (JSON)
79	Order 79 Data (JSON)

Key-Value: Useful Application Types

- Shopping Cart Data, typically with the key being the User ID
- Session Data, with the Session ID as the key
- User Profiles, with the User Name as the key
- User Preferences, with the User Name as the key

Key-Value: Less Suitable Situations

- When there is a requirement to correlate data stored with different sets of keys.
- When there is a need to wrap multiple key saves into a transaction. When a key-store operation fails, only that operation is undone – there is no provision to undo previous operations that may have been part of the same business transaction.
- When there is a requirement to handle sets of data within a single operation, such as updating all orders for a customer when the customer has defaulted on payments. Key-value database can retrieve only one key-value pair per option.

Document Databases

- Store and retrieve documents, which are typically formatted using JSON, XML, BSON, or a similar language. Like key-value databases, records are stored using a key and a value, with the value.
  - Unlike name-value databases, document databases are capable of parsing document contents, and many of them offer commands for searching document contents.
- While the documents in a given database typically have a common structure, the individual documents need not have identical structures.
  - It is also possible for attributes to have different names in different records.

Popular Document Databases

- MongoDB
- CouchDB
- Terrastore
- OrientDB
- RavenDB
- Lotus Notes

Document: Useful Application Types

- Event Logging, such as user login and updates of sales leads
- Content Management, such as user comments, blogs, user registrations, and user or customer profiles
- Web or Real-Time Analytics such as web click-streams and page views
- E-Commerce Applications that handle offering products or services and managing orders.

Document: Less Suitable Situations

- When requirements include complex transactions involving multiple documents.
  - However, some products such as RavenDB support complex transactions
- When requirements call for queries against varying document structures.
  - In particular, when the document structure varies from record to record, such as profiles with different data elements at different points in time, queries against the document parts that vary will not work well.

Columnar Databases

- Data organized into rows and columns, like relational. However, columns can be split into column groups (column families in Cassandra).
  - Column groups stored independently
  - Column group has a row key to facilitate re-assembly of complete rows
- Potentially significant performance improvement over relational because relational DBMSs must scan entire rows to extract the requested column data
- Most store columns as name-value pairs, which permits missing columns to be omitted.

Columnar Example

Customer Contact			
rowKey: 1	contactFirstName: Francisco	contactLastName: Pérez-Olaeta	jobTitle: Purchasing Manager
rowKey: 2	contactFirstName: Run	contactLastName: Liu	jobTitle: Accounting Assistant

Customer Data Column Group				
rowKey: 1	customerID: 6	companyName: Company F	City: Milwaukee	State: WI
rowKey: 2	customerID: 20	companyName: Company Z	City: Miami	State: FL

Popular Columnar Databases

- Cassandra
- Hbase
- Hypertable
- Amazon SimpleDB
- HP Vertica
  - Some argue that Vertica does not fit NoSQL because it uses standard SQL (with extensions) for all data operations.

Columnar: Useful Application Types

- Event Logging, such as user login and updates of sales leads
- Content Management, such as user comments, blogs, user registrations, and user or customer profiles
- HP Vertica has many other use cases because it supports SQL.

Columnar: Less Suitable Situations

- Full ACID transaction support for writes and reads
- Column family design changes are expensive in Cassandra. Therefore, it is not a great fit for early prototypes of applications where rapid changes are the norm.
  - Cassandra is better suited to applications where the query patterns are well established because query pattern changes usually lead to column family design changes.

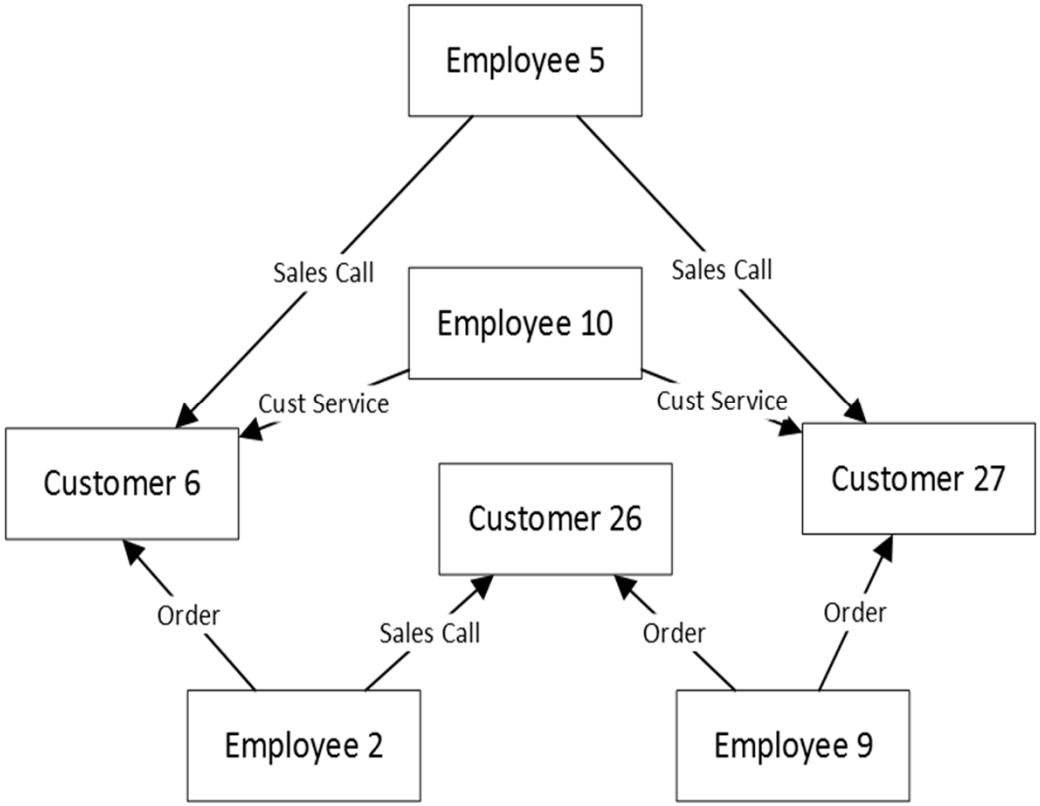
Graph Databases

- Graph databases store entities as nodes and attributes as properties of the nodes.
- Relationships between nodes are stored as edges, and new edges may be created as needed.
  - Unlike relational databases, adding edges (relationships) does not require adding properties (foreign key attributes) to nodes (entities).
  - Can be queried by specifying properties and edges
- Address complaints that relational databases are not well suited to graphically displaying data relationships.
- Structure and use quite different compared to other NoSQL databases.

Popular Graph Databases

- Neo4J
- Infinite Graph
- OrientDB
- FlockDB

Graph DB Example: Relationships Between Customers and Employees



Graph: Useful Application Types

- Connected data, such as showing diseases, treatments, and treatment side-effects.
- Routing, dispatch and location-based services, such as finding stores near a user's current location which also have a particular product in stock.
- Recommendation engines, such as finding movie recommendations based on viewing history, or products that are usually bought together.

Graph: Less Suitable Situations

- Any requirement to apply mass updates for all or a subset of entities (nodes) when attribute values change in the source data. Updating properties for existing nodes is generally not optimal.
- For very large volumes, operations that involve the entire graph may not perform well.