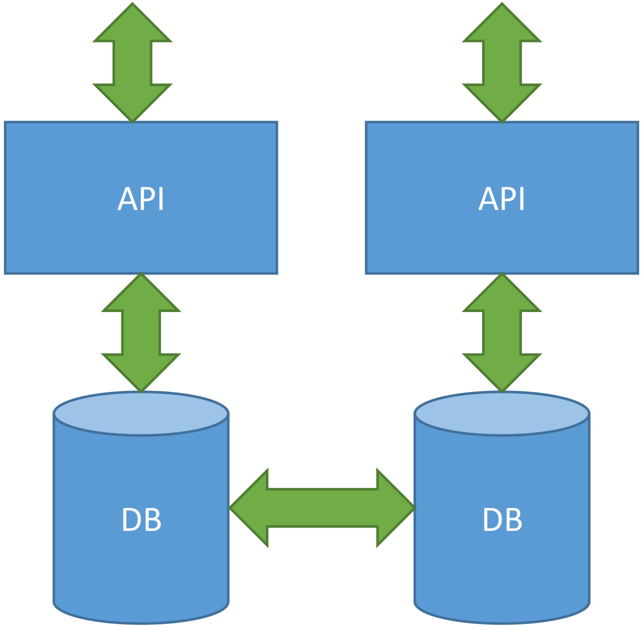
BLUEMETAL
A division of Insight

# Polyglot Persistence in Azure

Ken Seier

# Why Polyglot Persistence

- Polyglot Persistence is storing data in multiple stores, often with different formats
- Term coined in 2006
- So much more than microservices
- It's about maintaining performance for low total cost of ownership (TCO) *in large systems*



# Performance hockey stick

# Familiar Polyglot pattern

### Index Table

| Secondary Key (Town) | Customer Reference (ID) |
|---|---|
| Chicago | ID: 5 |
| Chicago | ID: 9 |
| Chicago | ID: 1000 |
| ... | ... |
| Portland | ID: 3 |
| Portland | ID: 7 |
| Redmond | ID: 1 |
| Redmond | ID: 4 |
| Redmond | ID: 6 |
| Redmond | ID: 8 |
| Seattle | ID: 2 |
| ... | ... |

### Fact Table

| Primary Key (Customer ID) | Customer Data |
|---|---|
| 1 | LastName: Smith, Town: Redmond, ... |
| 2 | LastName: Jones, Town: Seattle, ... |
| 3 | LastName: Robinson, Town: Portland, ... |
| 4 | LastName: Brown, Town: Redmond, ... |
| 5 | LastName: Smith, Town: Chicago, ... |
| 6 | LastName: Green, Town: Redmond, ... |
| 7 | LastName: Clarke, Town: Portland, ... |
| 8 | LastName: Smith, Town: Redmond, ... |
| 9 | LastName: Jones, Town: Chicago, ... |
| ... | ... |
| 1000 | LastName: Clarke, Town: Chicago, ... |
| ... | ... |

### Index Table

| Secondary Key (LastName) | Customer Reference (ID) |
|---|---|
| Brown | ID: 4 |
| Clarke | ID: 7 |
| Clarke | ID: 1000 |
| Green | ID: 6 |
| Jones | ID: 2 |
| Jones | ID: 9 |
| ... | ... |
| Robinson | ID: 3 |
| Smith | ID: 1 |
| Smith | ID: 5 |
| Smith | ID: 8 |
| ... | ... |

# Polyglot challenges

- Data consistency
- Transactional integrity
- Implementation complexity
- Creates data silos

# Workloads

# Transactional workloads

- Have clearly defined use patterns
- Focused on create, read, update, delete (CRUD)
- Affect a very limited volume of data, ideally single record
- Ideally write-few/read-few but often write-few/read-many

# Slicing workloads

- Have clearly defined use patterns
- Covers large numbers of records
- Slices (sub-selects) and aggregates data
- Commonly used for reporting and metrics
- Can be significantly optimized with indexing and caching

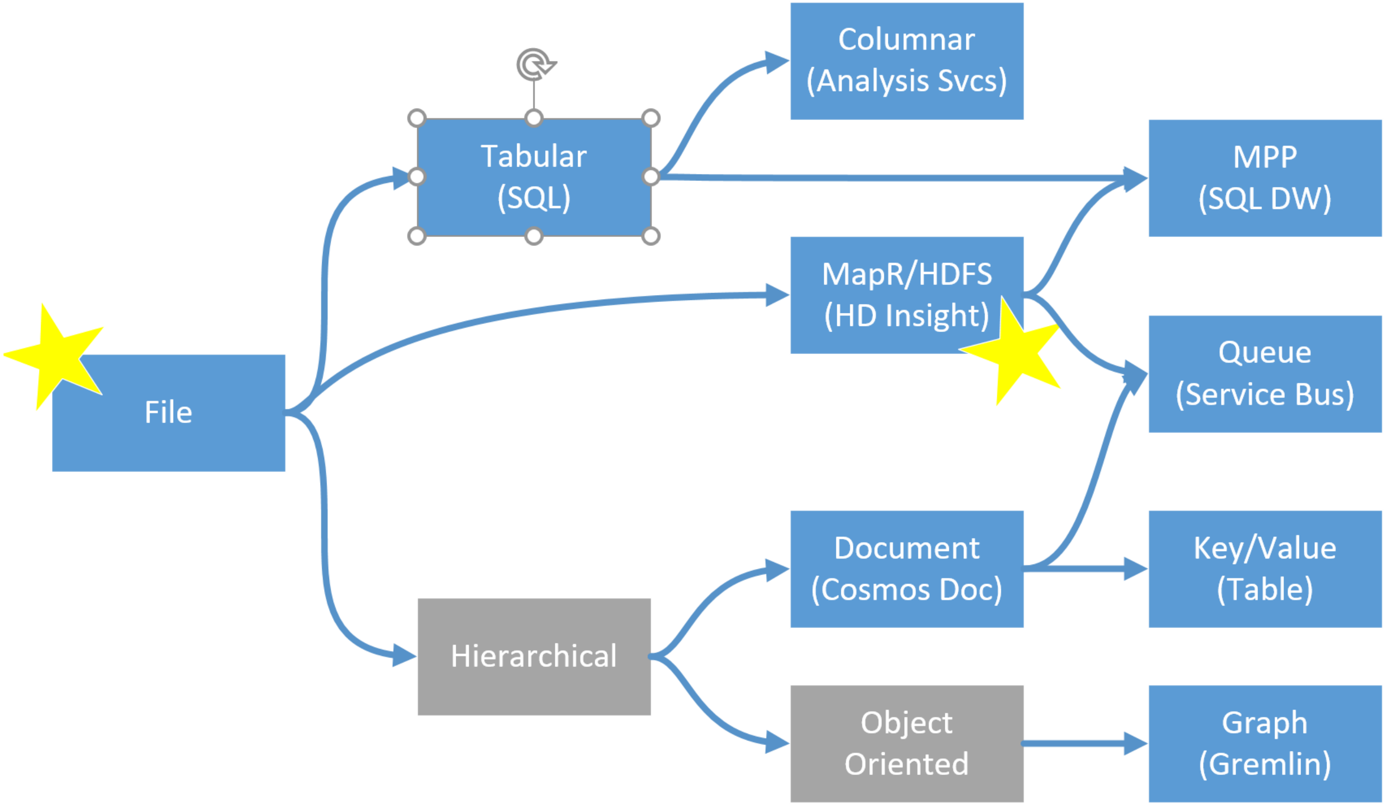
# Exploratory workloads

- Has no defined pattern
- Often cover very large amounts of data
- Thwarts optimization

# Different types of workloads in a single store

# Data Stores

# Data store family tree



# Many choices

- Driven by:
  - Backwards combability
  - Available skill
  - Future strategy



BLUEMETAL
A division of Insight

# Tabular stores

**STORE**

| Store_key | City | Region |
|---|---|---|
| 1 | New York | **East** |
| 2 | Chicago | Central |
| 3 | Atlanta | **East** |
| 4 | Los Angeles | **West** |
| 5 | San Francisco | **West** |
| 6 | Philadelphia | **East** |
| . | . | . |
| . | . | . |

**PRODUCT**

| Product_key | Description | Brand |
|---|---|---|
| 1 | Beautiful Girls | MKF Studios |
| 2 | Toy Story | **Wolf** |
| 3 | Sense and Sensibility | Parabuster Inc. |
| 4 | Holiday of the Year | **Wolf** |
| 5 | Pulp Fiction | MKF Studios |
| 6 | The Juror | MKF Studios |
| 7 | From Dusk Till Dawn | Parabuster Inc. |
| 8 | Hellraiser: Bloodline | Big Studios |
| . | . | . |
| . | . | . |

**SALES_FACT**

| Store_key | Product_key | Sales | Cost | Profit |
|---|---|---|---|---|
| 1 | 6 | 2.39 | 1.15 | 1.24 |
| 1 | 2 | 16.7 | 6.91 | 9.79 |
| 2 | 7 | 7.16 | 2.75 | 4.40 |
| 3 | 2 | 4.77 | 1.84 | 2.93 |
| 5 | 3 | 11.93 | 4.59 | 7.34 |
| 5 | 1 | 14.31 | 5.51 | 8.80 |
| . | . | . | . | . |
| . | . | . | . | . |

- Stores data in rectangular tables of rows with a fixed set of columns
- Tables are related via keys
- Enables strong normalization (deduplication) of data
- Good for transactional and slicing workloads
- OK for exploratory workloads

# Columnar stores

- Stores data in tables of columns with each column fully indexed
- Relies heavily on memory and caching for performance
- Excellent at slicing workloads
- OK at transactional workloads
- Poor at exploratory workloads

| Record # | Name | Address | City | State |
|---|---|---|---|---|
| 0003623 | ABC | 125 N Way | Cityville | PA |
| 0003626 | Newburg | 1300 Forest Dr | Troy | VT |
| 0003647 | Flotsam | Industrial Pkwy | Springfield | MT |
| 0003705 | Jolly | 529 S 5th St. | Anywhere | NY |

# Document stores

```
{
 "Name": "Yogurt Depot",
 "Id":  1,
 "Revenue": 2000,
 "Cost": 100,
 "Category": [ "dessert", "food", "yogurt" ],
 "Visits": [
     {
         "day": "Mon",
         "visit_count": 300
     },
     {
         "day": "Tue",
         "visit_count": 700
     }
 ],
```

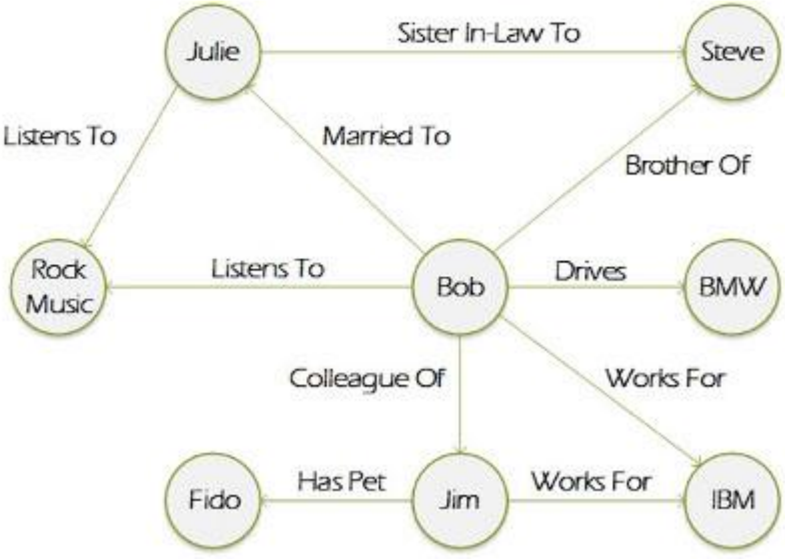
- Document defines a single record
- Document can contain just about any configuration of text-based data (JSON)
- Excellent at single-document transactional workloads
- Poor at slicing and exploratory workloads

# Key/value stores

- Stores binary blob value referenced by a unique key
- Blobs can be anything, but are often ragged column arrays
- Data often partitioned by key
- A smart key can improve performance by acting as an index
- Excellent at single record transactional workloads
- Poor at slicing and exploratory workloads

| Key | Value |
|---|---|
| K1 | AAA,BBB,CCC |
| K2 | AAA,BBB |
| K3 | AAA,DDD |
| K4 | AAA,2,01/01/2015 |
| K5 | 3,ZZZ,5623 |

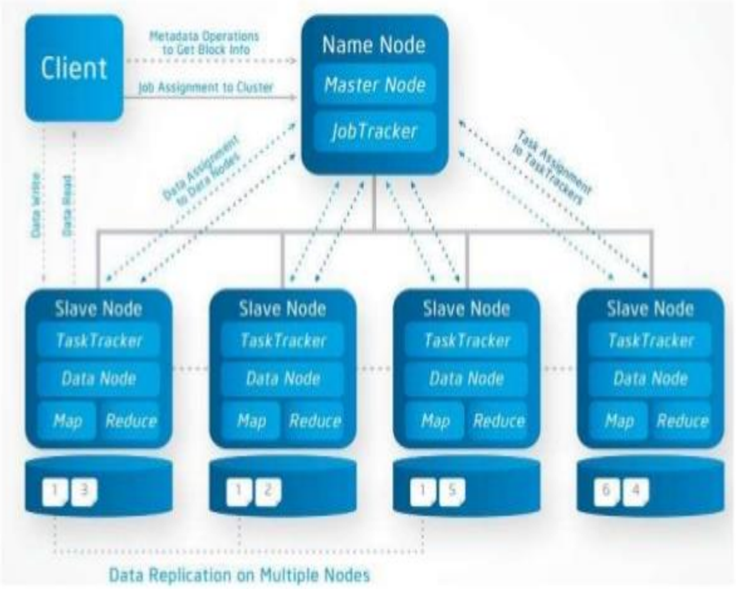
# Graph stores




- Stores data as nodes and node relationship (edges)
- Excellent at network-type workloads
- Poor at all other workloads

# MapR/HDFS stores

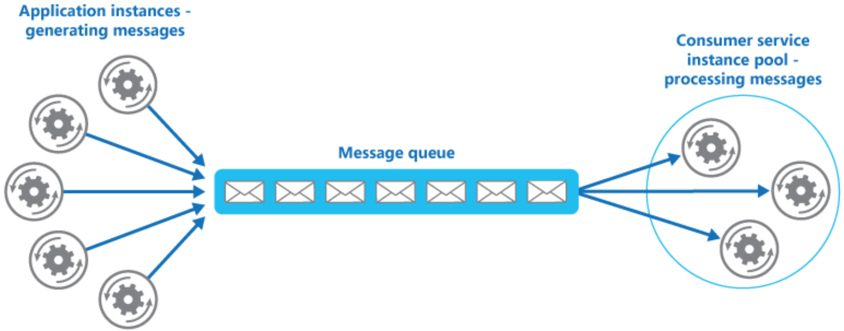
- Based on a “head” node controlling multiple “worker” nodes
- Can accept any type of binary data
- Extremely flexible
- Excellent at exploratory workloads
- Good at transactional and slicing workloads over large amounts of data




BLUEMETAL
A division of Insight

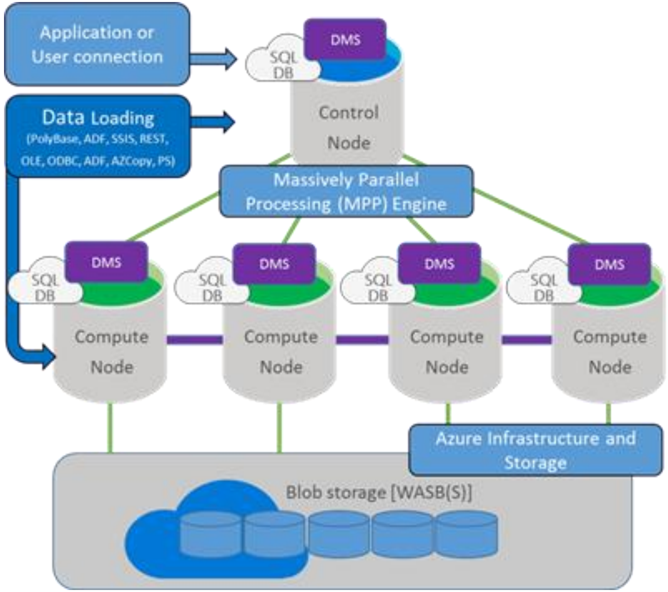
# Queue stores




- Stores messages (documents) in ordered format
- Optimized for ordered consumption messages
- Multiple patterns for handling messages: read-only, consume, reservation, etc.
- Excellent at some transactional workloads
- Terrible at slicing and exploratory workloads

# MPP stores

- Stores tabular data across multiple horizontal partitions (shards)
- Distributes large tables across shards using partition key
- Replicates small tables across shards
- Excellent at slicing workloads over large amounts of data
- OK at transactional workloads
- Poor at exploratory workloads

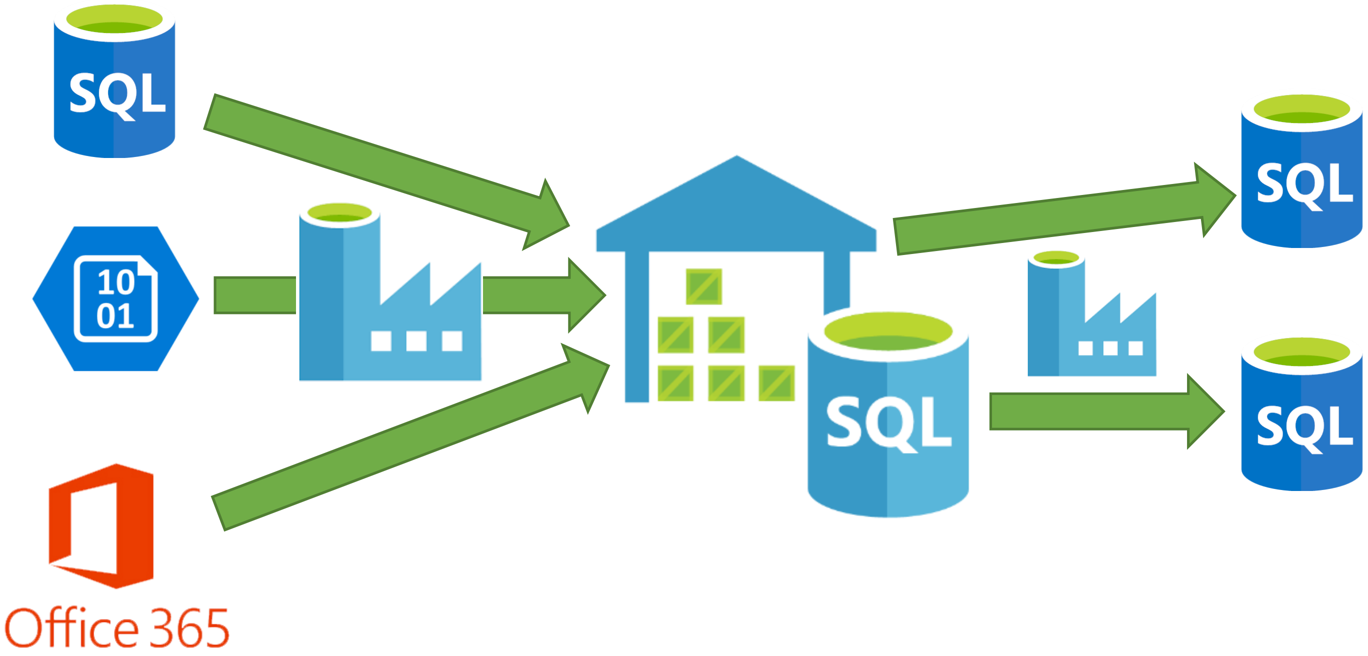




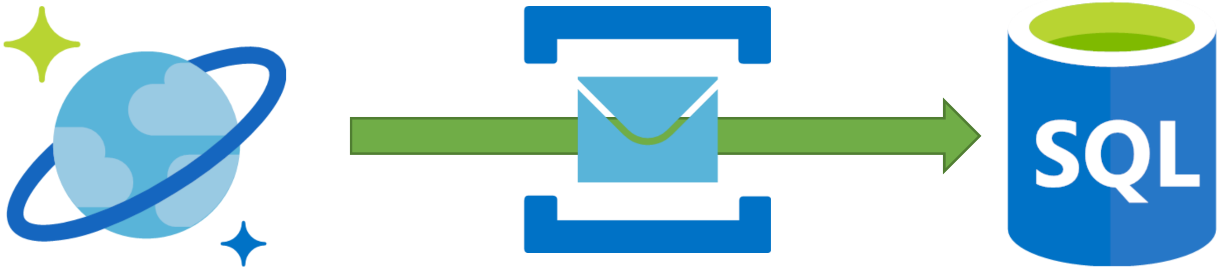
BLUEMETAL
A division of Insight

# Polyglot Patterns

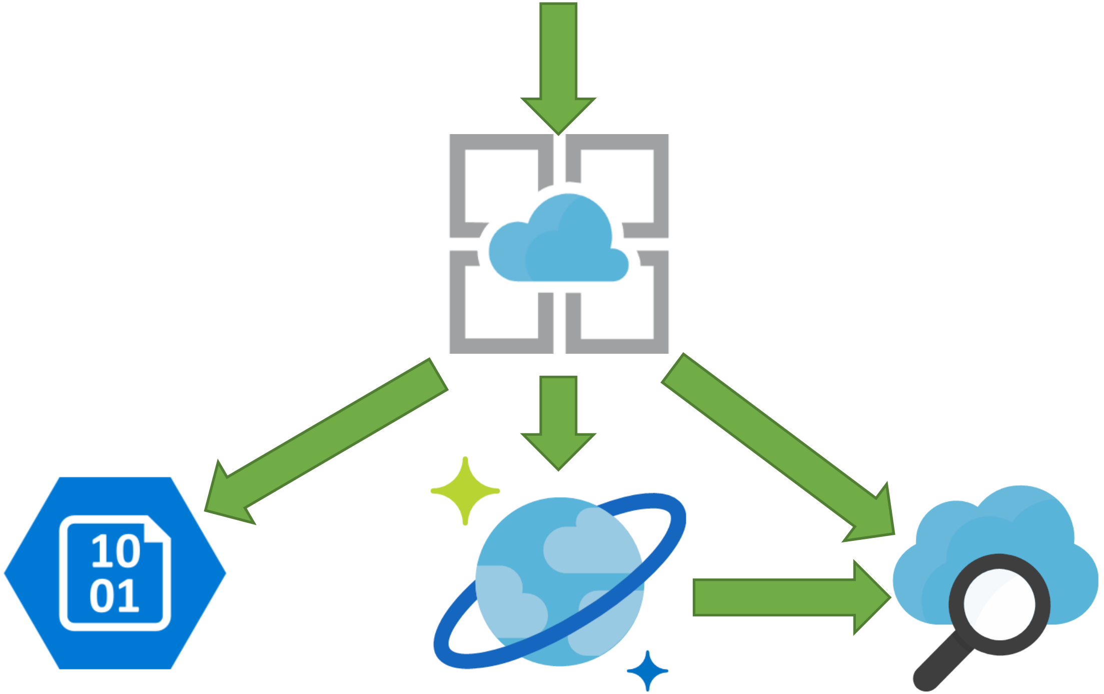
# OLTP/OLAP



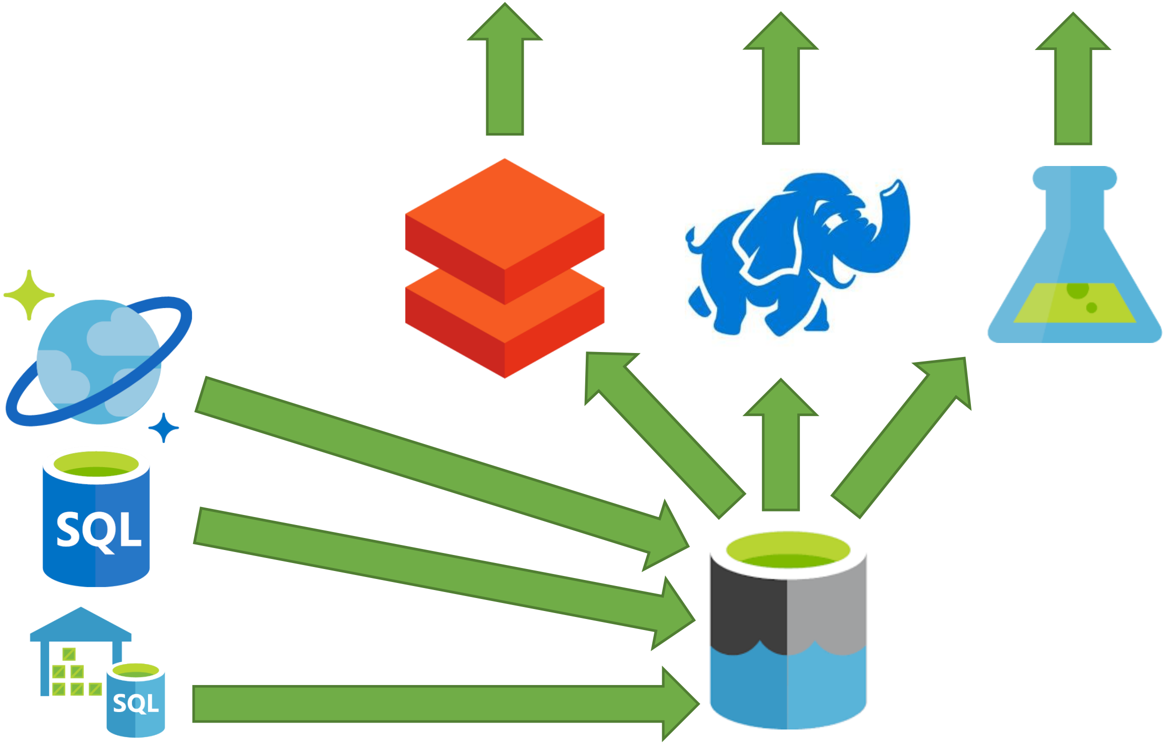
# Dependent systems



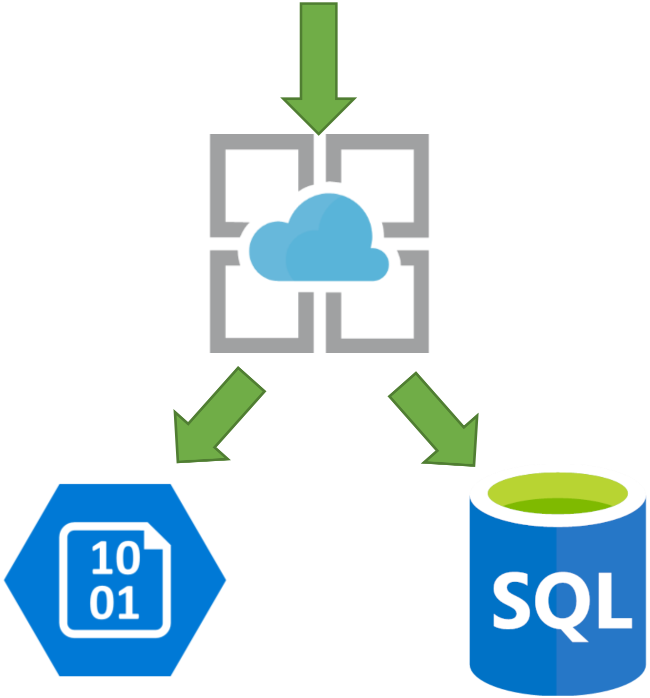
# Indexing systems



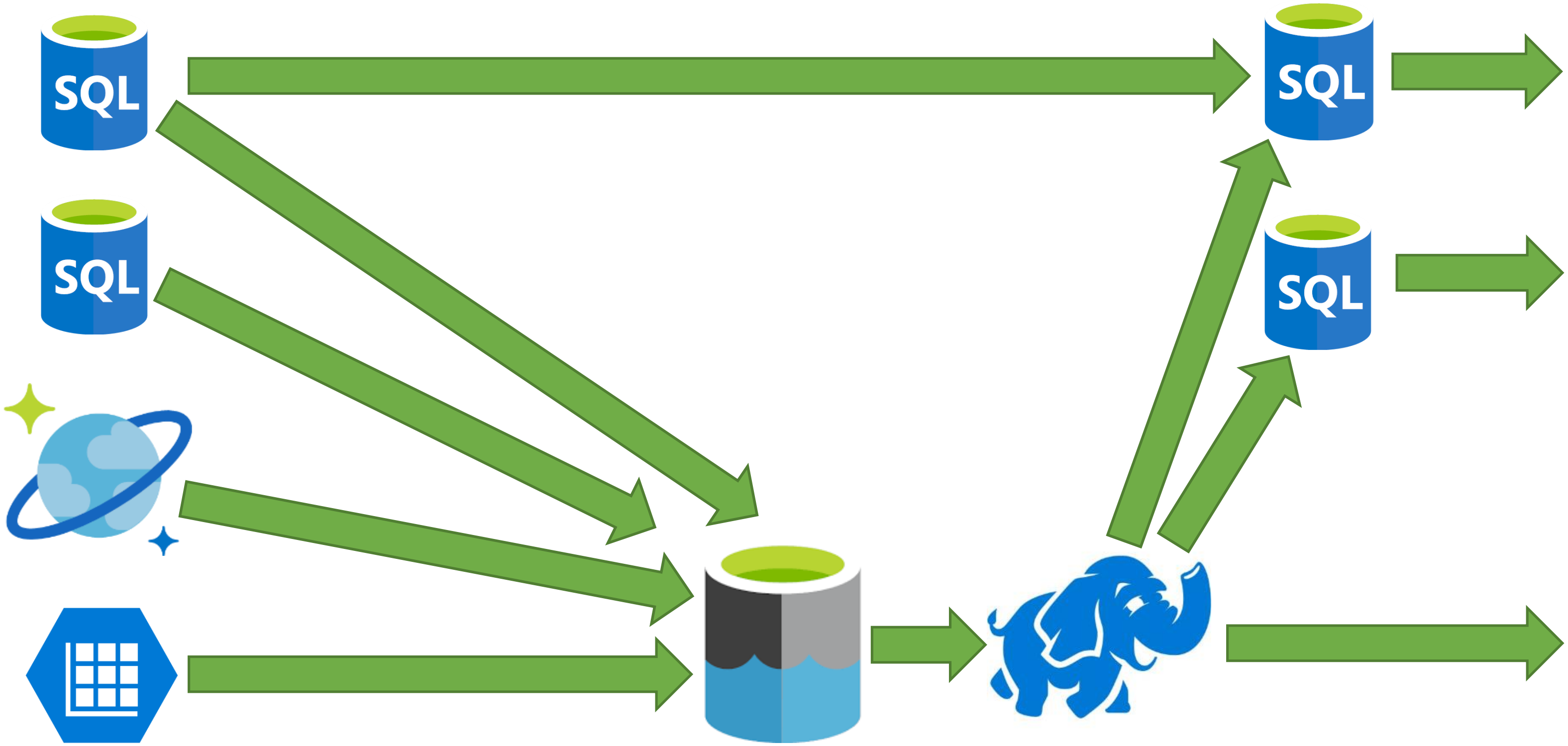
# Exploratory systems



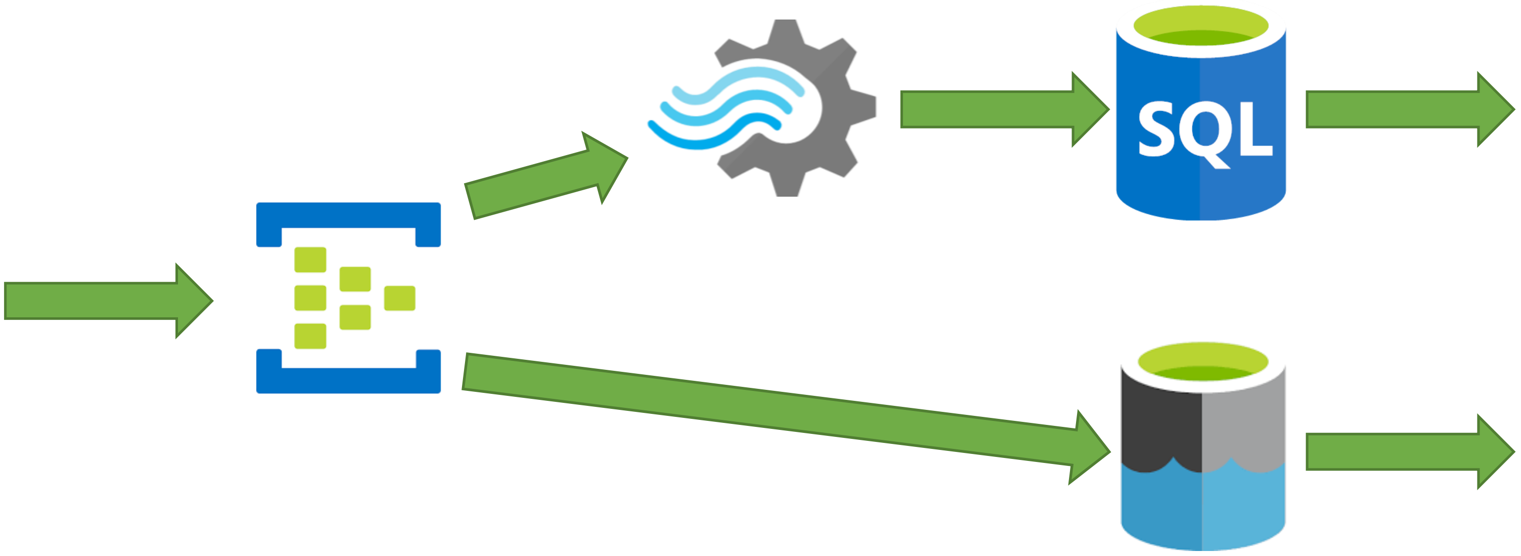
# Disparate data types



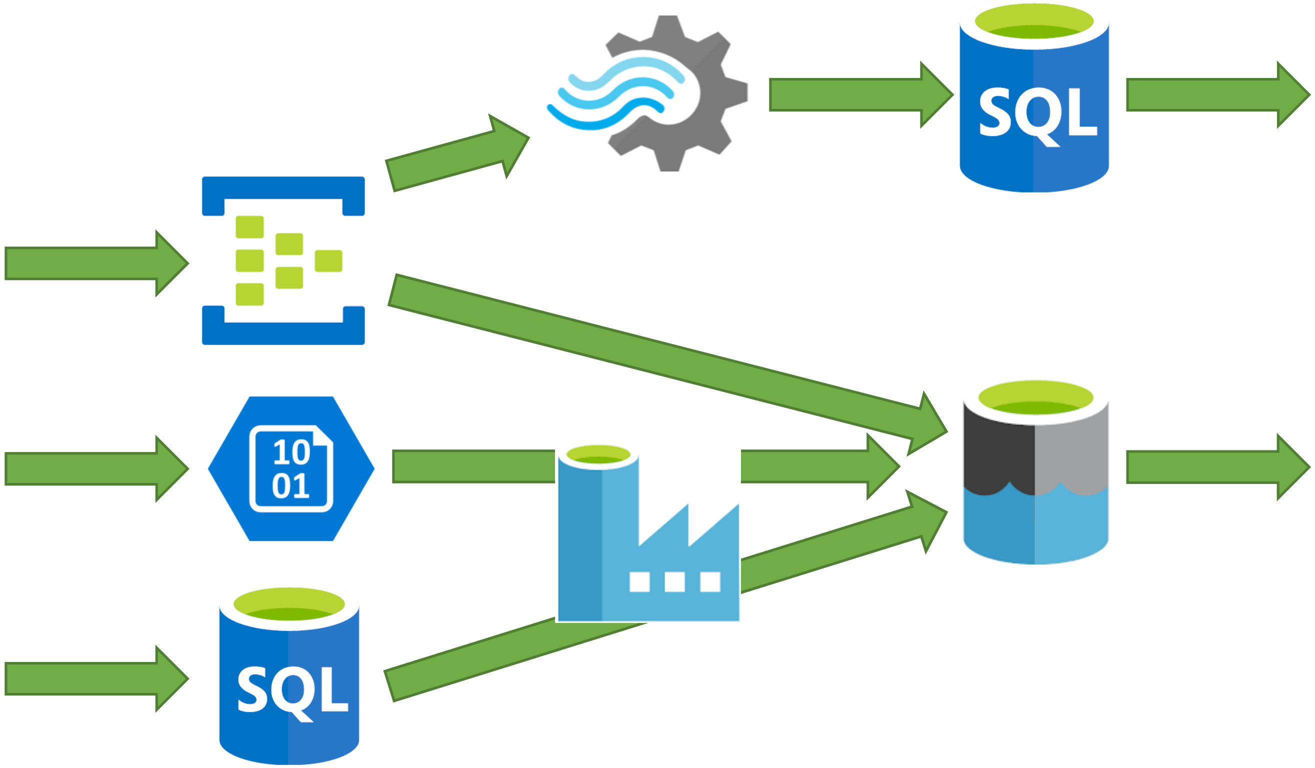
# Modern data warehouse



# Streaming systems (Kappa architecture)



# Lambda architecture



# Summary

# Bringing it together

- Polyglot persistence is aligning data stores with workloads to improve TCO. This can mean:
  - Performance costs
  - Platform/hardware/software costs
  - Development velocity
- Trades implementation complexity for TCO benefits
- Only needed in higher-complexity systems (large code base, large data, high performance needs, etc.)
- Lots of good choices, even more bad choices
- SQL isn't going anywhere

BLUEMETAL
A division of Insight

# Thank You

# A BIG thank you to the 2018 Global Sponsors!