

# DataStage & MongoDB

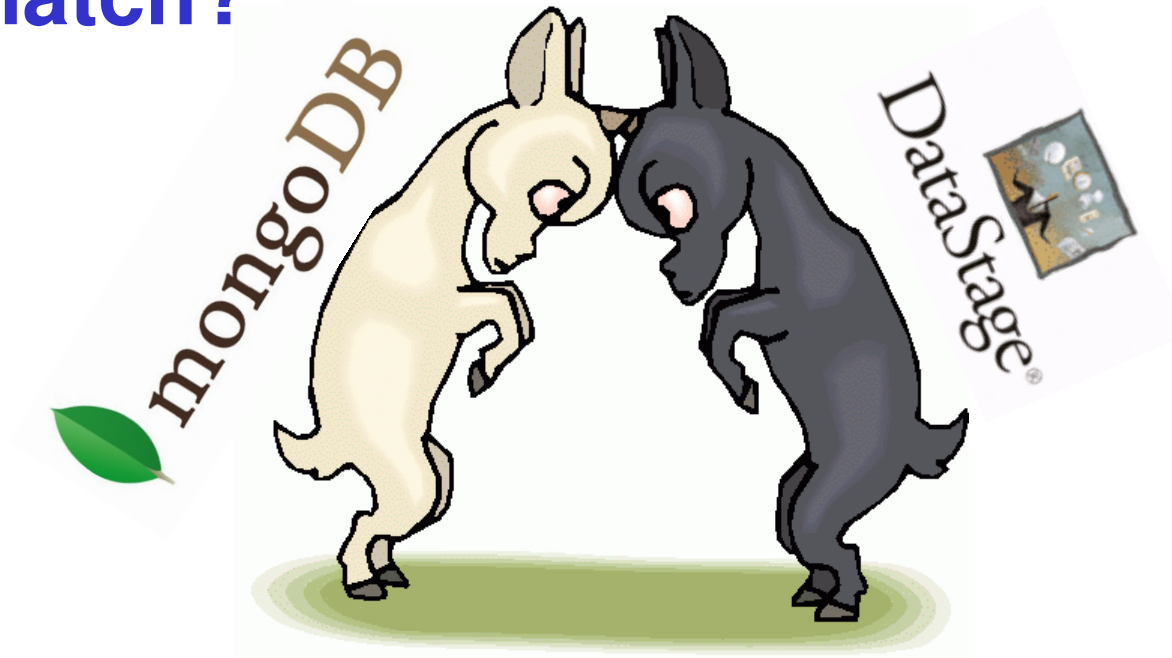
## NoSQL and ETL Collide!

Paul Stanley

IBM Information Server Connectivity Architect



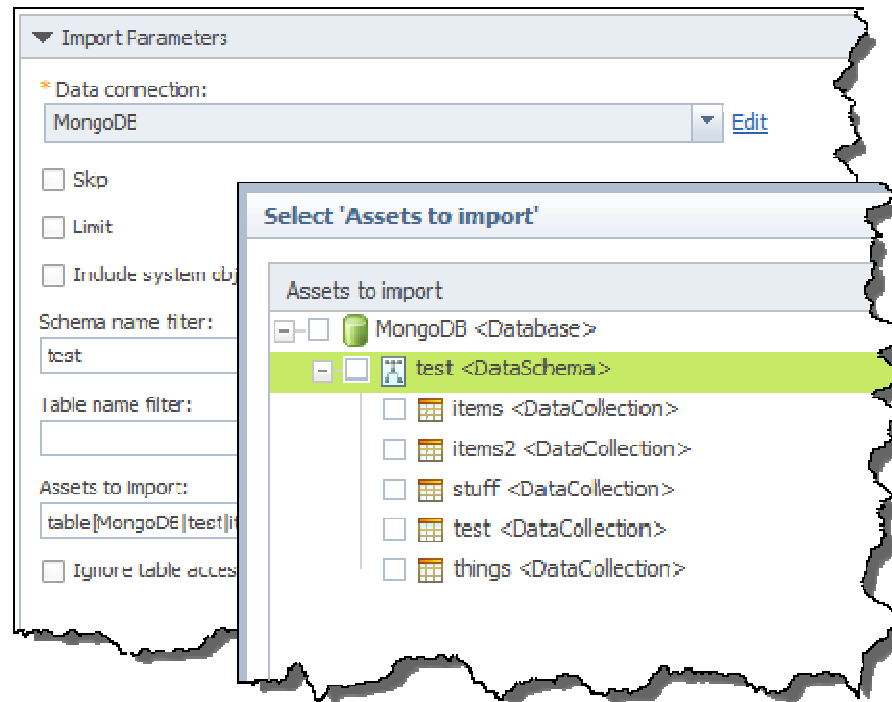
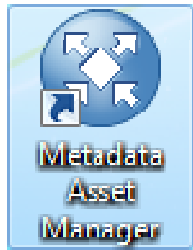
# NoSQL and ETL : An Impedance Mismatch?



- ETL is metadata driven
  - Transformation, governance, lineage
- NoSQL is metadata-free!



# Metadata Determined by Introspection

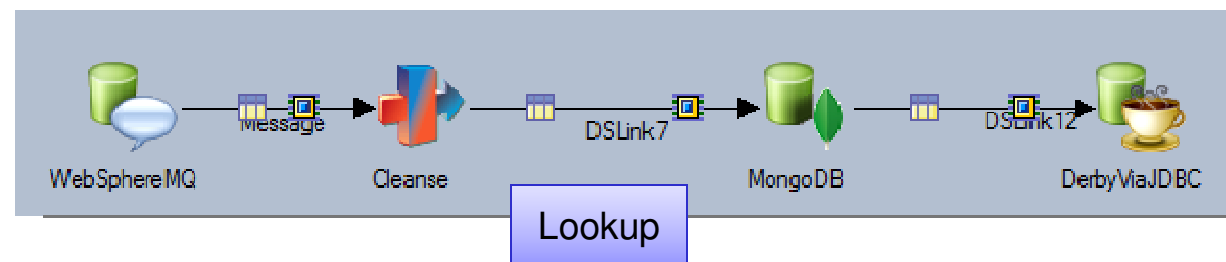
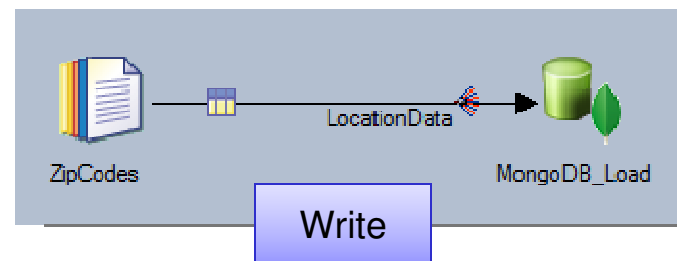
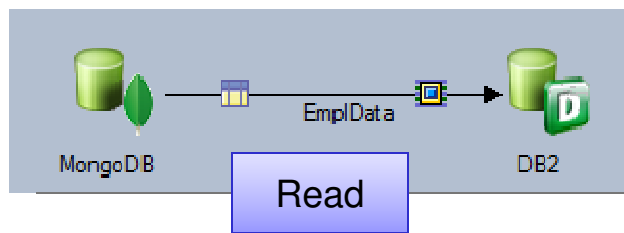


- Import metadata by examining a sample of documents
- Structure flattened to a relational view
- Metadata stored in Information Server repository



# MongoDB Connector Highlights

- Data transferred as “relational” or JSON, or a hybrid
- Create databases, create collections, create indexes
- Insert, Update, Replace or Delete
- Query using Mongo’s JSON syntax
- Sparse lookups
- Return metadata in jobs either as columns or JSON Schema
- Parallel reads



# MongoDB Connector Configuration

General		Properties	Advanced
▼ Connection			
Host	MONGO-SVR		
User name			
Password			
▼ Usage			
▼ Java			
Classpath	#CMongoDB.JAR#		
Database	Statistics		
Collection	ZipCodes		
Query expression	{flags.ChangedCase:{\$exists:true}}		
Sort	Population		
Skip	10		
Limit	30		
▶ RCP			
▼ Transfer J-SON			
JSON format	Yes		
Selection	Mongo		
Suppress ID	No		
Metadata mode	No		
Duplicates	No		

Read

General		Properties	Advanced
▼ Connection			
Host	MONGO-SVR		
User name			
Password			
▼ Usage			
▼ Java			
Classpath	#CMongoDB.JAR#		
Database	Statistics		
Collection	ZipCodes		
Write mode	Insert		
Database action	Create		
Collection action	Replace		
Index expression	Zipcode		
Query expression			
Transfer J-SON	No		

Write



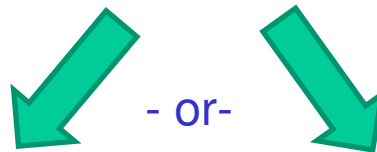
# MongoDB Array Handling\*

\* Coming soon...

```
{
  "type": "record",
  "fields":
  [
    {"name": "a", "type": "int"},
    {"name": "b", "type": {"type": "array", "items": "int"}}
  ]
}
```

- Primitive arrays and arrays of records can be denormalized.
- Supports nested arrays

```
[{ "a": 100, "b": [1001, 1002]}, { "a": 200, "b": [2001, 2002]}]
```

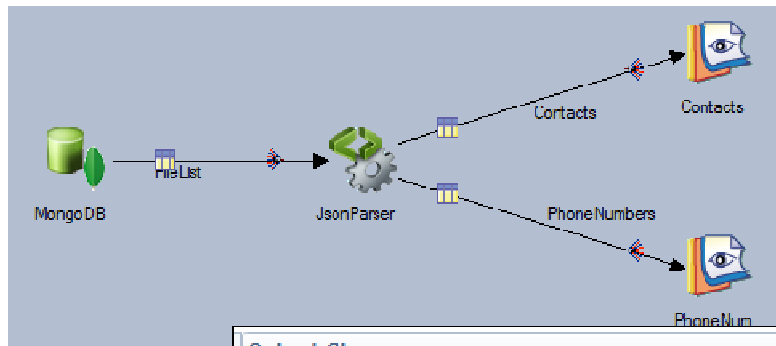


```
100 | [1001, 1002]
200 | [2001, 2002]
```

```
100 | 1001
100 | 1002
200 | 2001
200 | 2002
```



# MongoDB Connector + JSON Stage



- JSON Stage can be used to parse or compose JSON documents.
- Use with MongoDB Connector for complex document handling.

**Output Step**

Configuration Information Test Data

Output → Mappings

[Switch to Tree View](#)

Output Links: Contacts

☐ Enable Runtime Column Propagation

Name	Key	Type	Description/Path	Length	S...
firstName	<input type="checkbox"/>	VarChar	./JSON_Parser:result/ns0:root/firstName		
lastName	<input type="checkbox"/>	VarChar	./JSON_Parser:result/ns0:root/lastName		
age	<input type="checkbox"/>	VarChar	./JSON_Parser:result/ns0:root/age		
streetAddress	<input type="checkbox"/>	VarChar	./JSON_Parser:result/ns0:root/address/streetAddress		
city	<input type="checkbox"/>	VarChar	./JSON_Parser:result/ns0:root/address/city		
state	<input type="checkbox"/>	VarChar	./JSON_Parser:result/ns0:root/address/state		
postalCode	<input type="checkbox"/>	VarChar	./JSON_Parser:result/ns0:root/address/postalCode		
newSubscriptic	<input type="checkbox"/>	VarChar	./JSON_Parser:result/ns0:root/newSubscription		
companyName	<input type="checkbox"/>	VarChar	./JSON_Parser:result/ns0:root/companyName		
Click to add					

**Input**

Find:

- top
  - InputLinks
    - FileList
      - JSON
        - JSON\_Parser:result
          - ns0:root
            - firstName
            - lastName
            - age
            - address
              - phoneNumbers
              - newSubscription
              - companyName
              - @type



Thank  
YOU

