In this lecture, we will discuss…

- Categories of NoSQL
- NoSQL vs. RDBMS

Categories of NoSQL – Key/Value

- Value can be String or JSON
- Key-value hash
- Solutions
  - Dynamo
  - Redis
  - Memcached

ID	Attributes
1234	John Doe
1235	{ "Name": "Godfather", "Genre": "Drama", "Actor": "Robert DeNiro", "Director": "Francis Ford Coppola" }

Categories of NoSQL – Document

- Stores documents based up of tagged elements
- Persistent and query-able
- Solutions
  - MongoDB
  - CouchDB

```
{
   "id": 1234,
   "name": "Departed",
   "actors": [
      {
         "actor": "Leonardo DeCaprio"
      },
      {
         "actor": "Jack Nicholson"
      }
   ],
   "director": "Martin Scorsese",
   "genre": "drama"
}
```

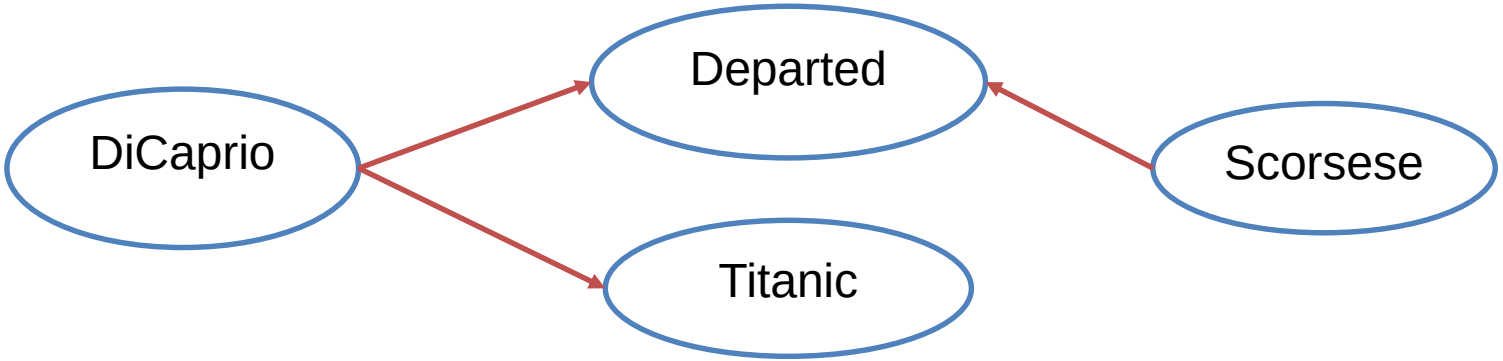
Categories of NoSQL – Column

- Uses flat structure, but with keys stored in columns rather than rows:
- Solutions
  - Cassandra
  - Hbase

ID	101	102	103
Name	The Godfather	The Departed	Titanic
Actor	Leonardo DiCaprio	Al Pacino	Leonardo DiCaprio
Director	Francis Ford Coppola	Martin Scorsese	James Cameron

Categories of NoSQL – Graph

- A network database that uses edges and nodes to represent and store data
- Solutions
  - Neo4J



NoSQL – Not supported

- ✧ Joins are not supported
  - Embedded documents or in middle tier code
- ✧ ACID Transactions
  - Supported at a document level only

NoSQL vs RDBMS – How to pick?

- Nature of data
  - Row/column (structured) – RDBMS
  - Unstructured, complex (geo-spatial or engineering data) which needs nesting - NoSQL
- Schema
  - Static – RDBMS, Dynamic – NoSQL

NoSQL vs RDBMS – How to pick?

- Self contained – NoSQL, Joins – RDBMS
- Flexibility of query
  - RDBMS Joins allow for flexibility
  - NoSQL - Duplication of data, implement joins in middle-ware

Summary

- 4 different categories offering different choices
- Pick what is best for your application (Relational or NoSQL)

What’s Next?

MongoDB