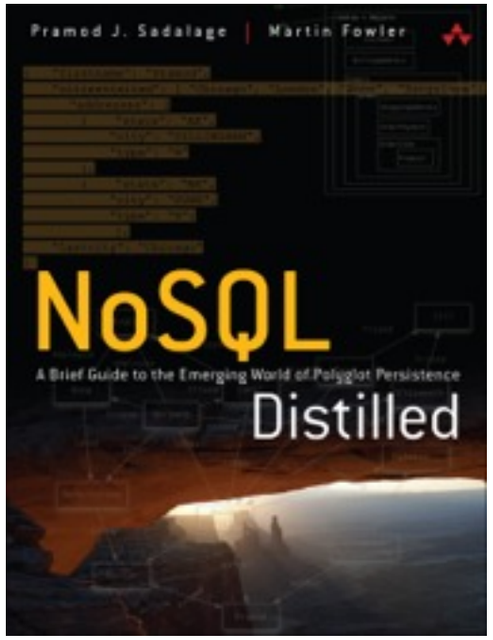


NoSQL Distilled

A guide to polyglot persistence

#NoSQLDistilled
@pramodsadalage
ThoughtWorks Inc.

Why RDBMS?

ACID Transactions

Atomicity
Consistency
Isolation
Durability

Standard Query Interface

Interact with many languages

Everyone knows SQL

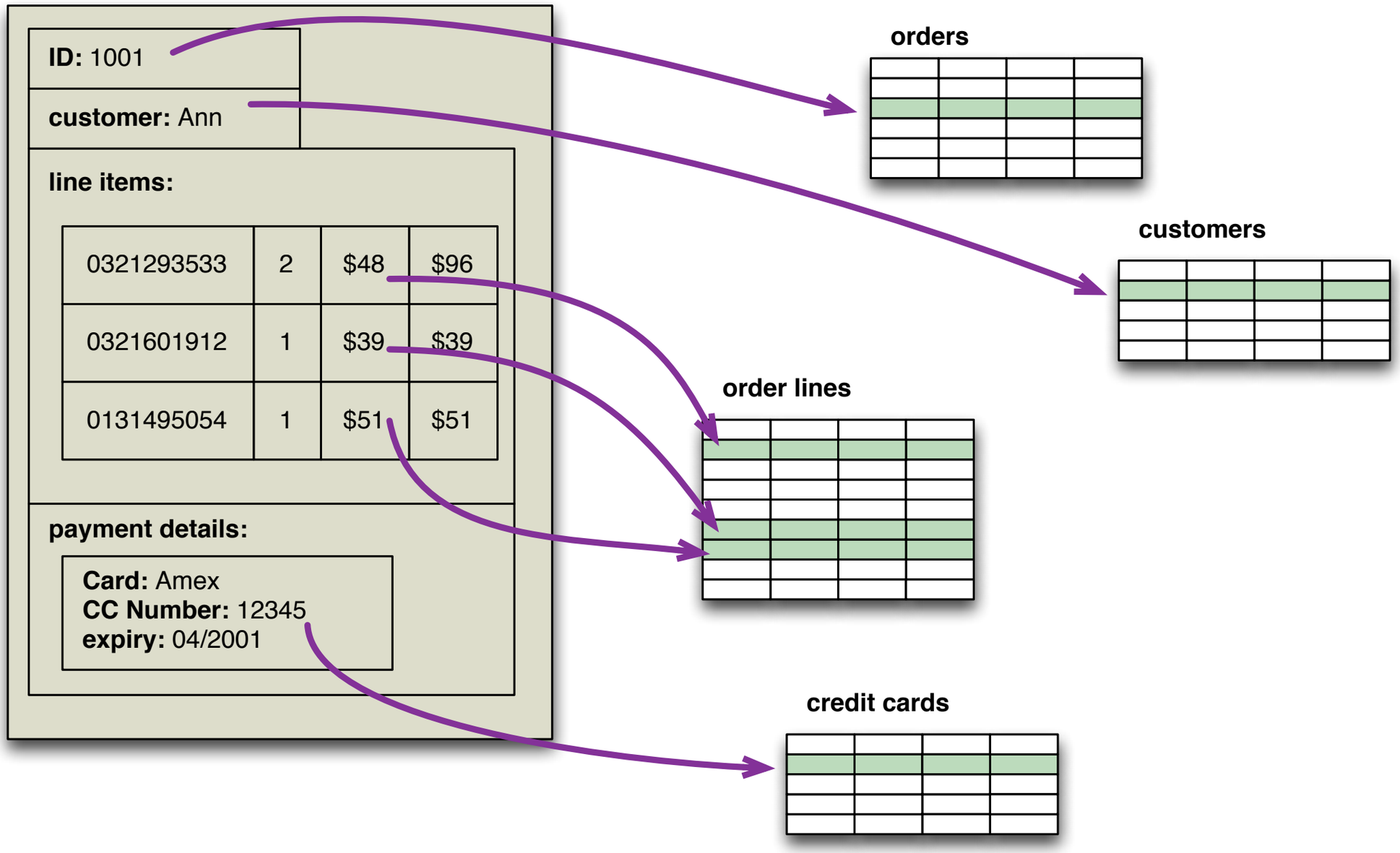
Everyone knows SQL

Limit less indexing

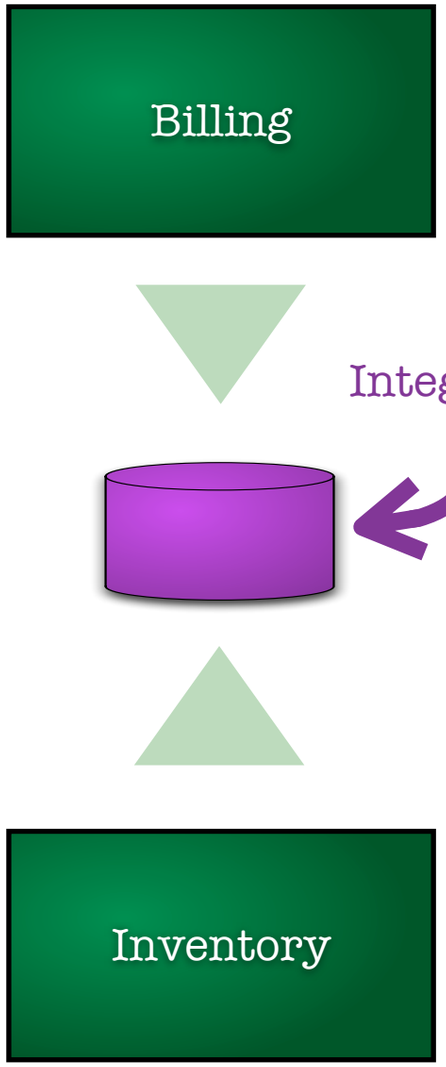
Handles many data models

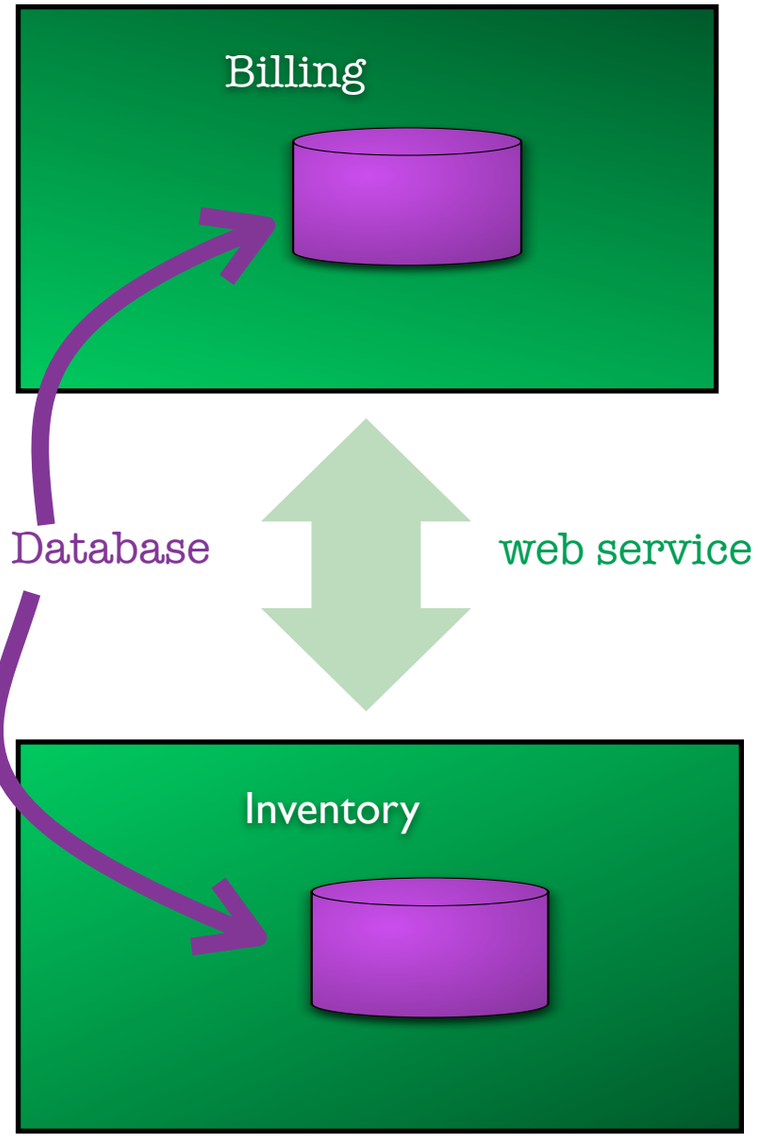
Why NoSQL

Schema changes are hard



Impedance mismatch





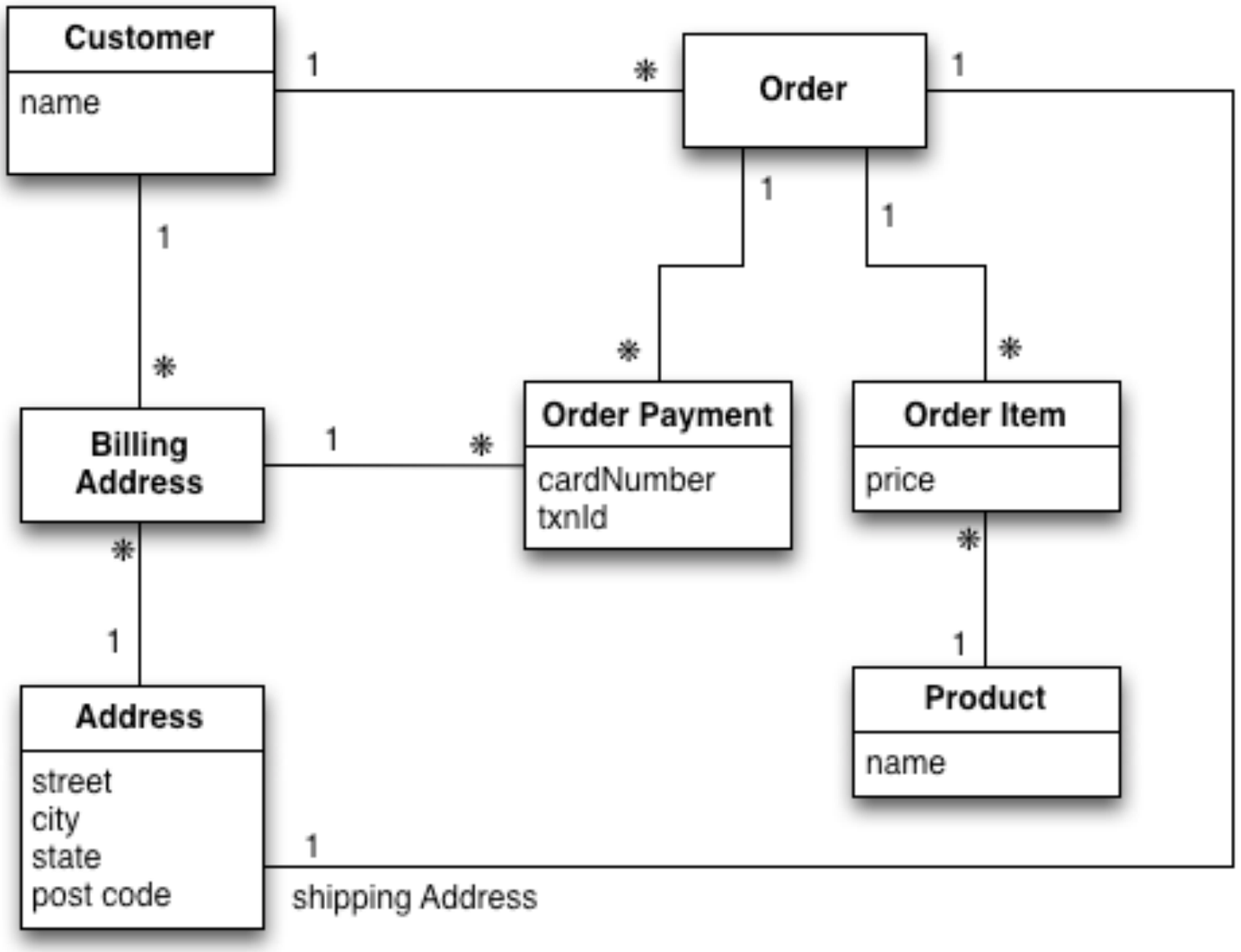
Application vs Integration databases

Running on clusters

Un-Structured Data

Un-Even rate of data growth

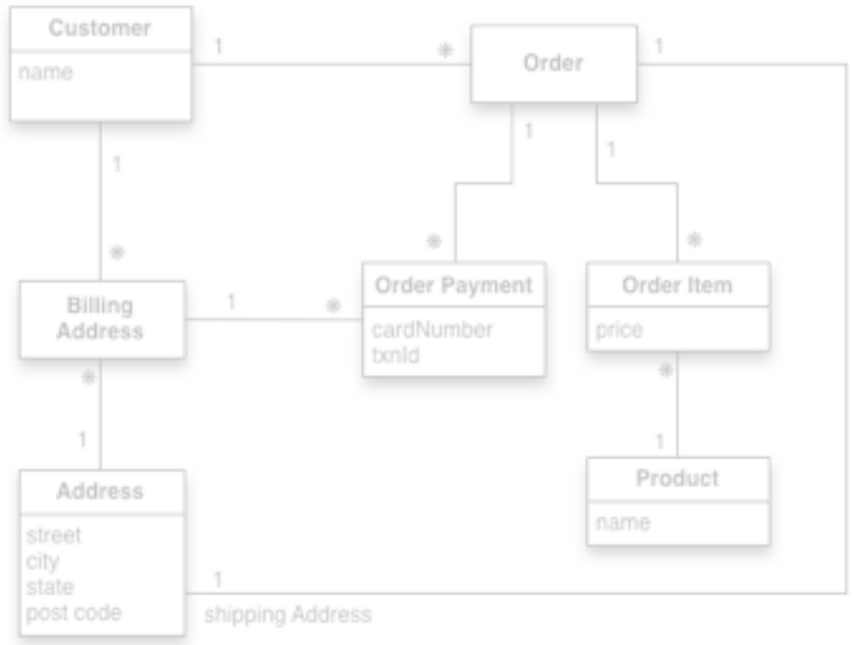
Domain Models



Domain driven data models

Customer	
Id	**Name**
1	Martin

Orders		
Id	**CustomerId**	**ShippingAddressId**
99	1	77



Product	
Id	**Name**
27	NoSQL Distilled

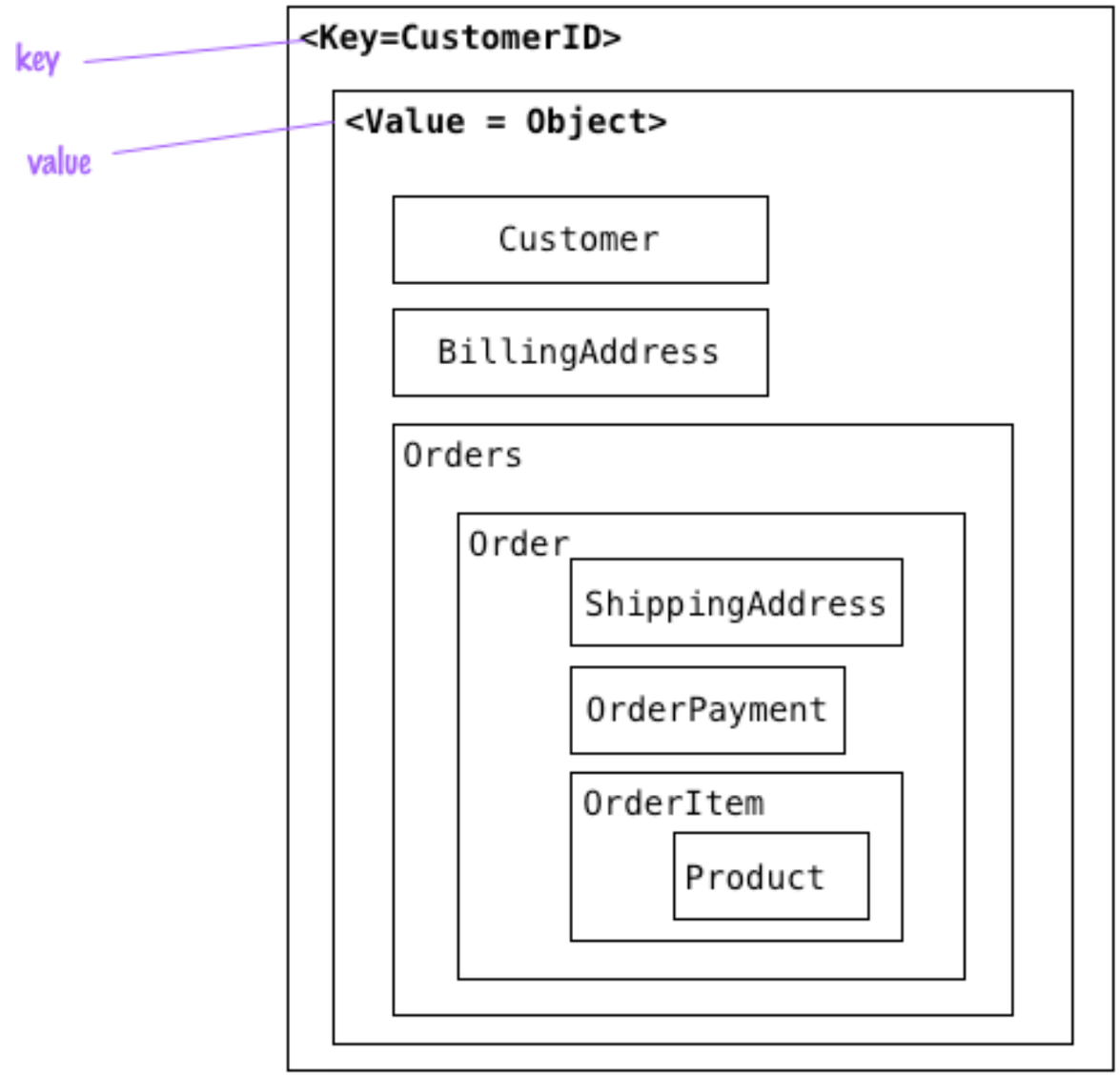
BillingAddress		
Id	**CustomerId**	**AddressId**
55	1	77

OrderItem			
Id	**OrderId**	**ProductId**	**Price**
100	99	27	32.45

Address	
Id	**City**
77	Chicago

OrderPayment				
Id	**OrderId**	**CardNumber**	**BillingAddressId**	**txnId**
33	99	1000-1000	55	abelif879rft

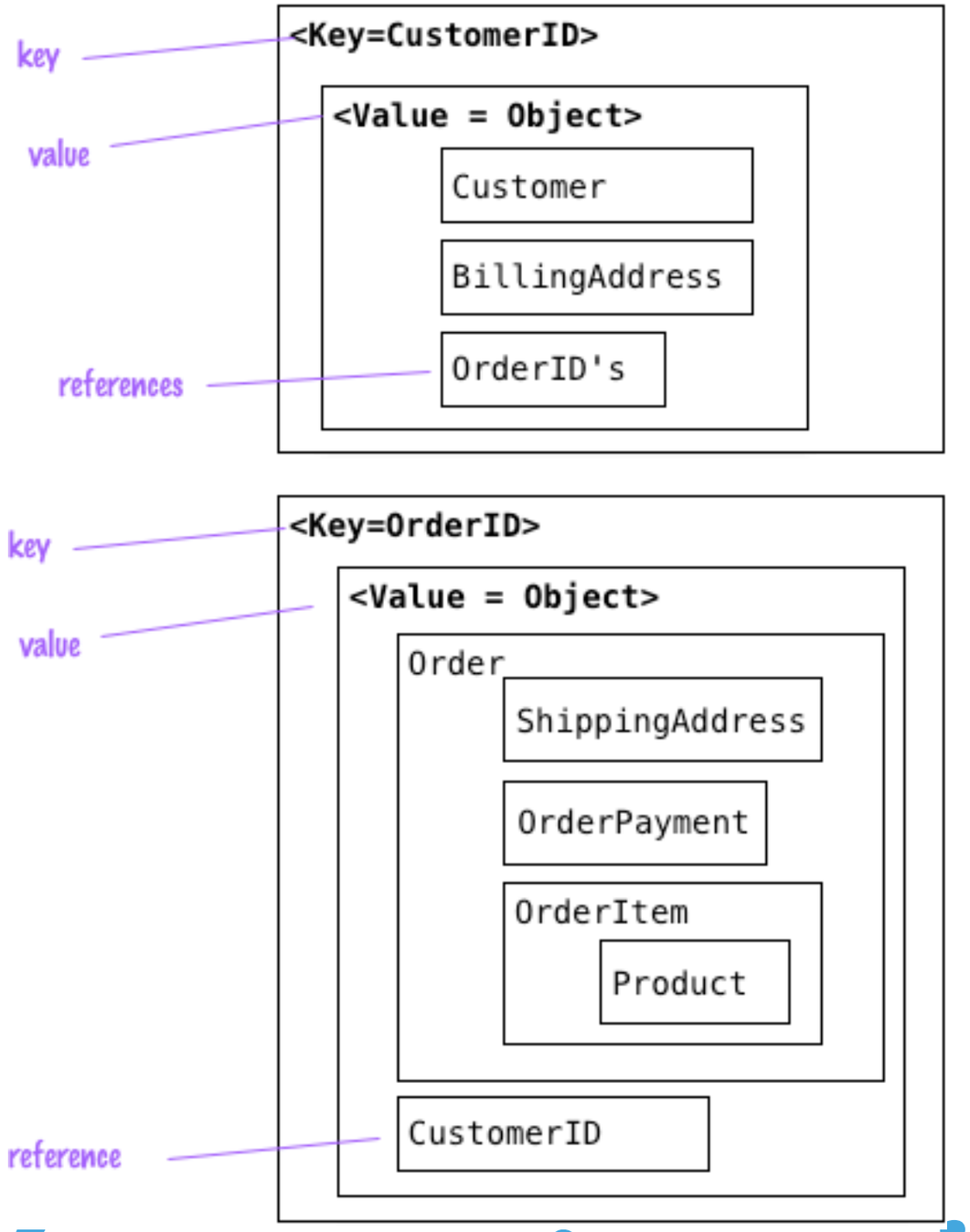
RDBMS data



Aggregate model (Embedding objects)

```
    // in customers
{
  "customer": {
    "id": 1,
    "name": "Martin",
    "billingAddress": [{"city": "Chicago"}],
    "orders": [
        {
          "id":99,
          "orderItems":[
            {
              "productId":27,
              "price": 32.45,
              "productName": "NoSQL Distilled"
            }
          ],
          "shippingAddress":[{"city":"Chicago"}]
          "orderPayment":[
            {
              "ccinfo":"1000-1000-1000-1000",
              "txnId":"abelif879rft",
              "billingAddress": {"city": "Chicago"}
            }
          ],
        }
    ]
}}
```

Aggregate Data



Aggregate model (Referencing Objects)

```
    // in Customers
{
   "id":1,
   "name":"Martin",
   "billingAddress":[{"city":"Chicago"}]
}
// in Orders
{
   "id":99,
   "customerId":1,
   "orderItems":[
     {
       "productId":27,
       "price": 32.45,
       "productName": "NoSQL Distilled"
     }
   ],
   "shippingAddress":[{"city":"Chicago"}]
   "orderPayment":[
     {
       "ccinfo":"1000-1000-1000-1000",
       "txnId":"abelif879rft",
       "billingAddress": {"city": "Chicago"}
     }
   ],
}
```

Aggregate data

Aggregate Orientation

RDBMS's have no concept of aggregates

Aggregates reduce the need for ACID

Better for clusters, can be distributed easily

Key-Value
Document
Column-Family

Key Value Databases

Key-Value Database

- One Key-One Value
- Value is opaque to database
- Like a Hash
- Some are distributed

Oracle	Riak
instance	cluster
table	bucket
row	key-value
row-id	key

riak

redis

Project Voldemort
A distributed database.

MEMCACHED

{ “key” (VIN) } = { “value” (car facts) }

JTTDR…

…
make#Ford
model#Mustang
year#2011
….

Document Databases

Document Database

- One Key-One Value
- Value is visible to database
- Value can be queries
- JSON/XML documents

Oracle	MongoDB
instance	mongod
schema	database
table	collection
row	document
row_id	_id

mongoDB

RAVEN DB

CouchDB relax

terrastore

OrientDB

{ “id” (VIN) } = { “document” (car facts) }

JTTDR…

= {…
“make”: “Ford”,
“model”: “Mustang”,
“year”: 2011,
…
}

Column-Family Databases

Column-Family Database

- Data organized as columns
- Each row has row key
- Columns have versioned data
- Row data is sorted by column name

Oracle	Cassandra
instance	cluster
database	keyspace
table	column-family
row	row
columns same for every row	columns can be different for each row

{ “id” (VIN) } = { “column families”(car facts) }

Cassandra

HYPERTABLE INC

APACHE HBASE

(JTTDR…) = ({…
“car”:{“make”:“Ford”,
“model”:“focus”…}
“service”:{…}
})

Key-Points Aggregate Databases

Inter-aggregate relations are hard to maintain

Schema-less means implicit schema

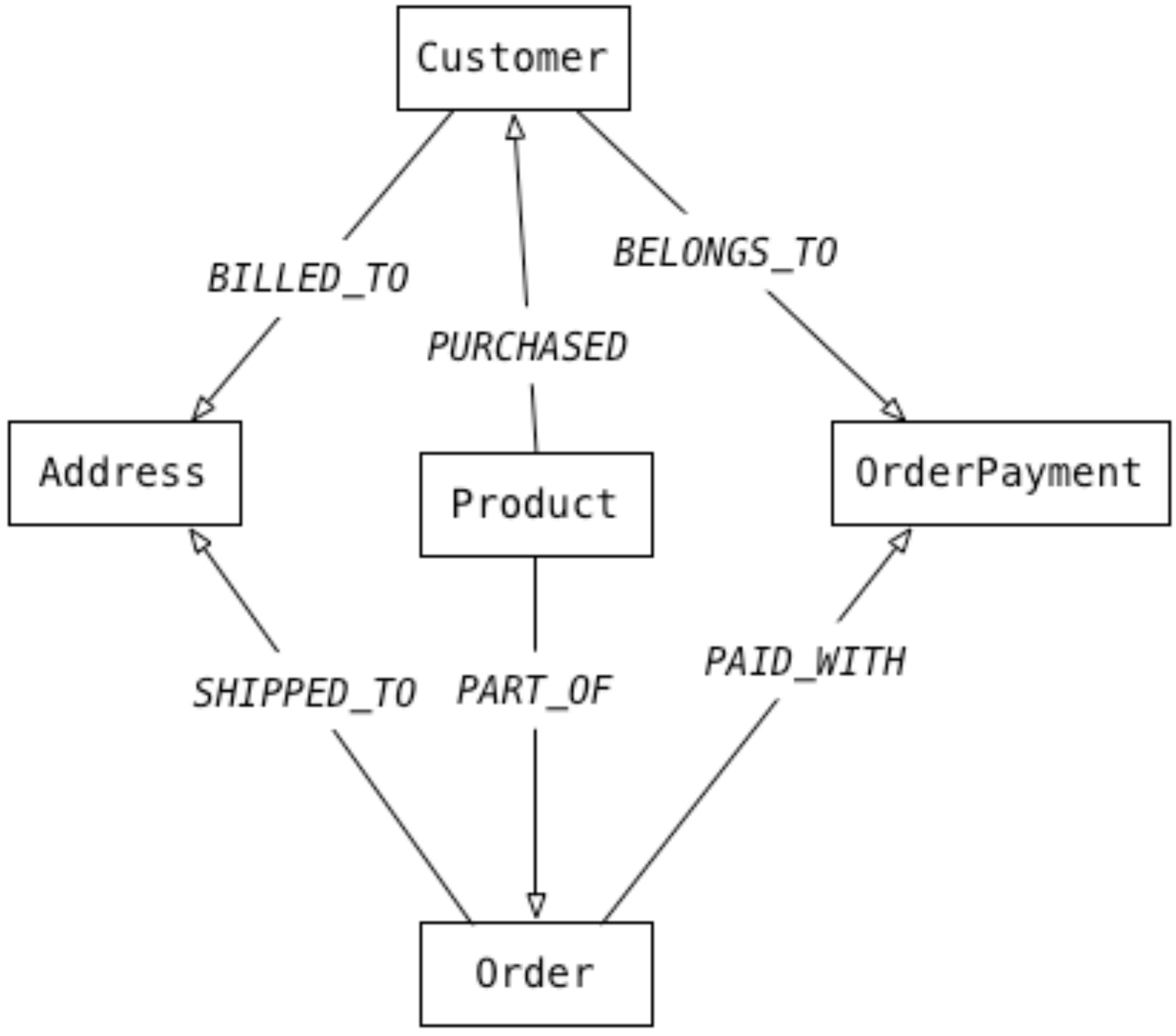
Graph Databases

Neo4j
NOSQL for the Enterprise

InfiniteGraph

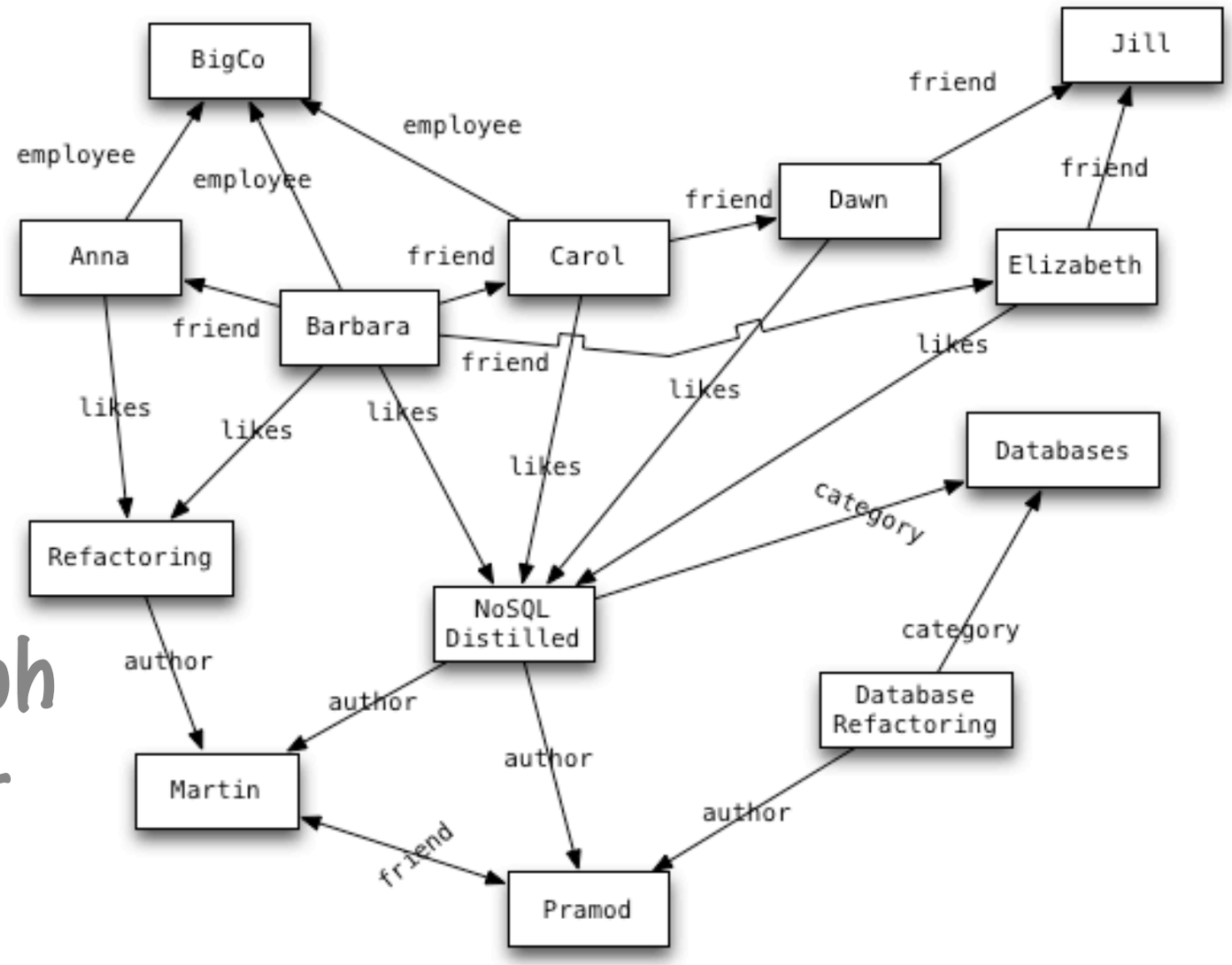
OrientDB

HYPERGRAPHDB



Graph Databases

- Is multi-relational graph
- Relationships are first-class citizens
- Traversal algorithms
- Nodes and Edges can have data (key-value pairs)



Graph databases work best for data with complex relations

Key-Value Database Usage

Session Storage

User Profiles/Preferences

Shopping Cart

Single user analytics

Document Database Usage

Event Logging

Prototype development

eCommerce Application

Content Management Applications

Column-Family Database Usage

Large write volume

Content Management

eCommerce Application

Graph Database Usage

Connected Data

Routing things/money

Location Services

Recommendation engines

Schema-less really?

Schema-free does not mean no schema-migration

Schema is implicit in code

Data must be migrated, when schema in code is changed

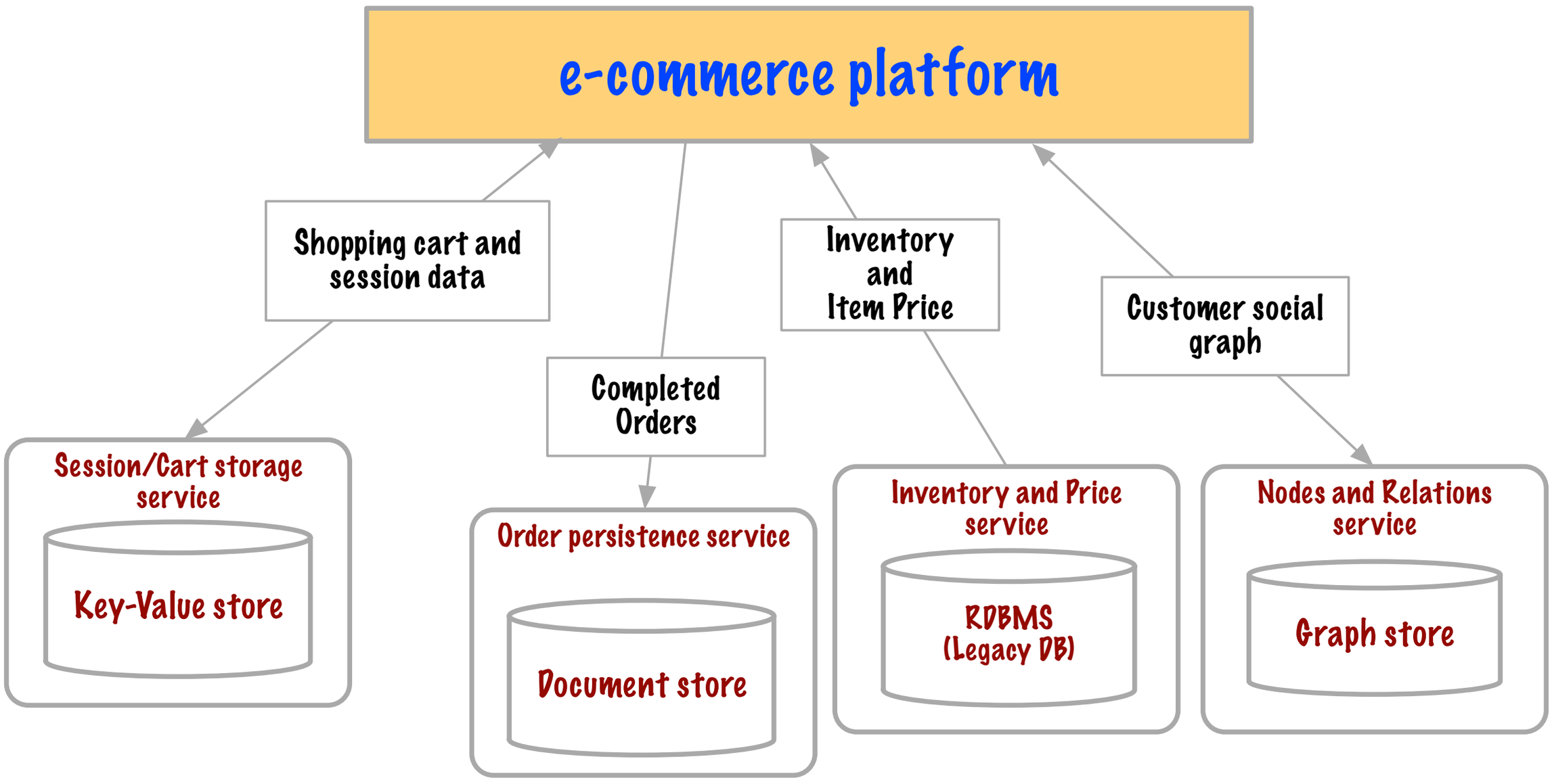
All data need not be migrated at the same time (lazy migration)

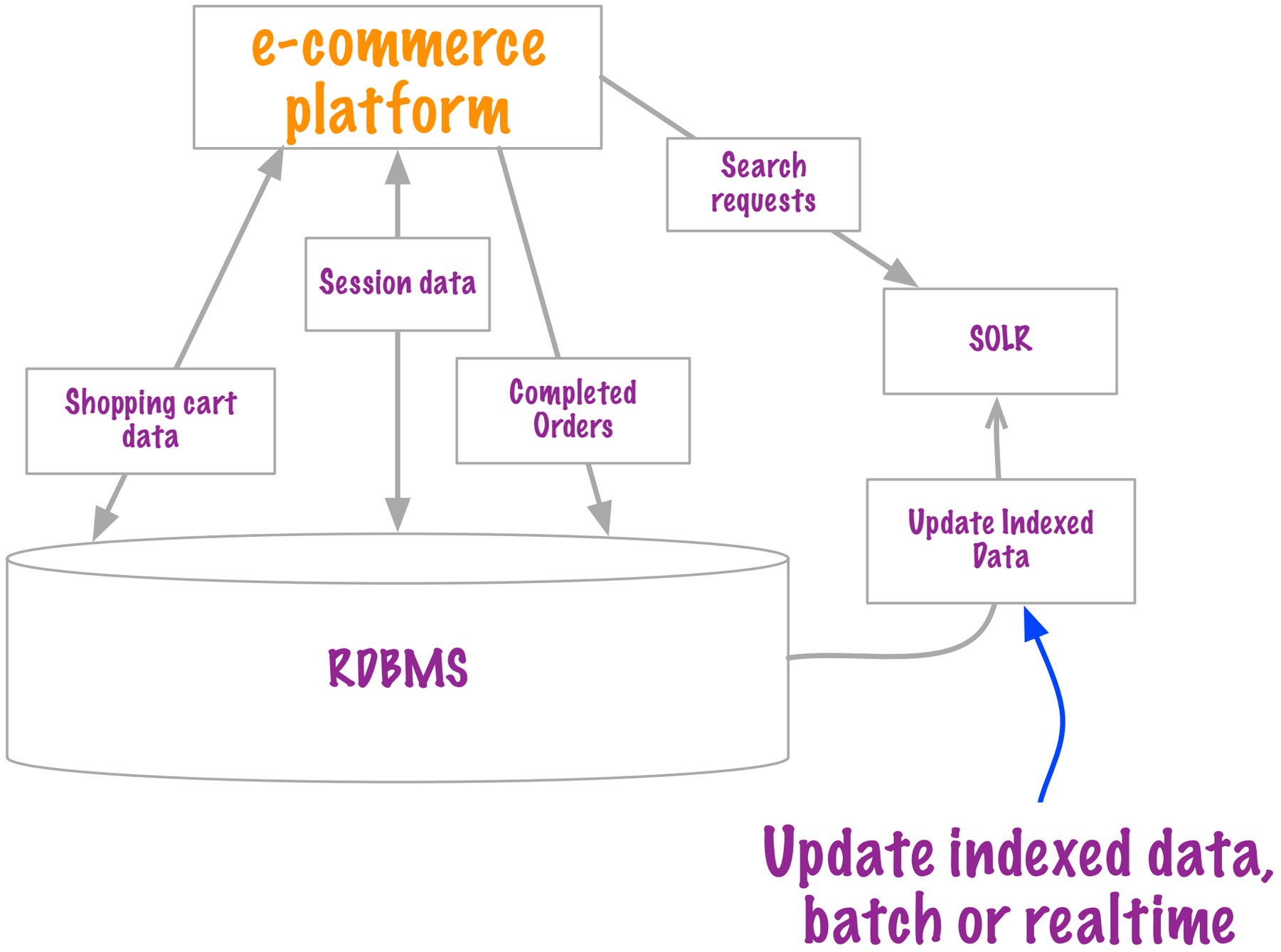
Polyglot Persistence

Use different data storage technology for varying needs

Can be across the enterprise or in single application

Encapsulate data access through services


e-commerce platform
Shopping cart and session data
Completed Orders
Inventory and Item Price
Customer social graph
Session/Cart storage service
Key-Value store
Order persistence service
Document store
Inventory and Price service
RDBMS (Legacy DB)
Nodes and Relations service
Graph store


e-commerce platform
Search requests
SOLR
Session data
Shopping cart data
Completed Orders
Update Indexed Data
RDBMS
Update indexed data, batch or realtime

Speculative Retail Web Application

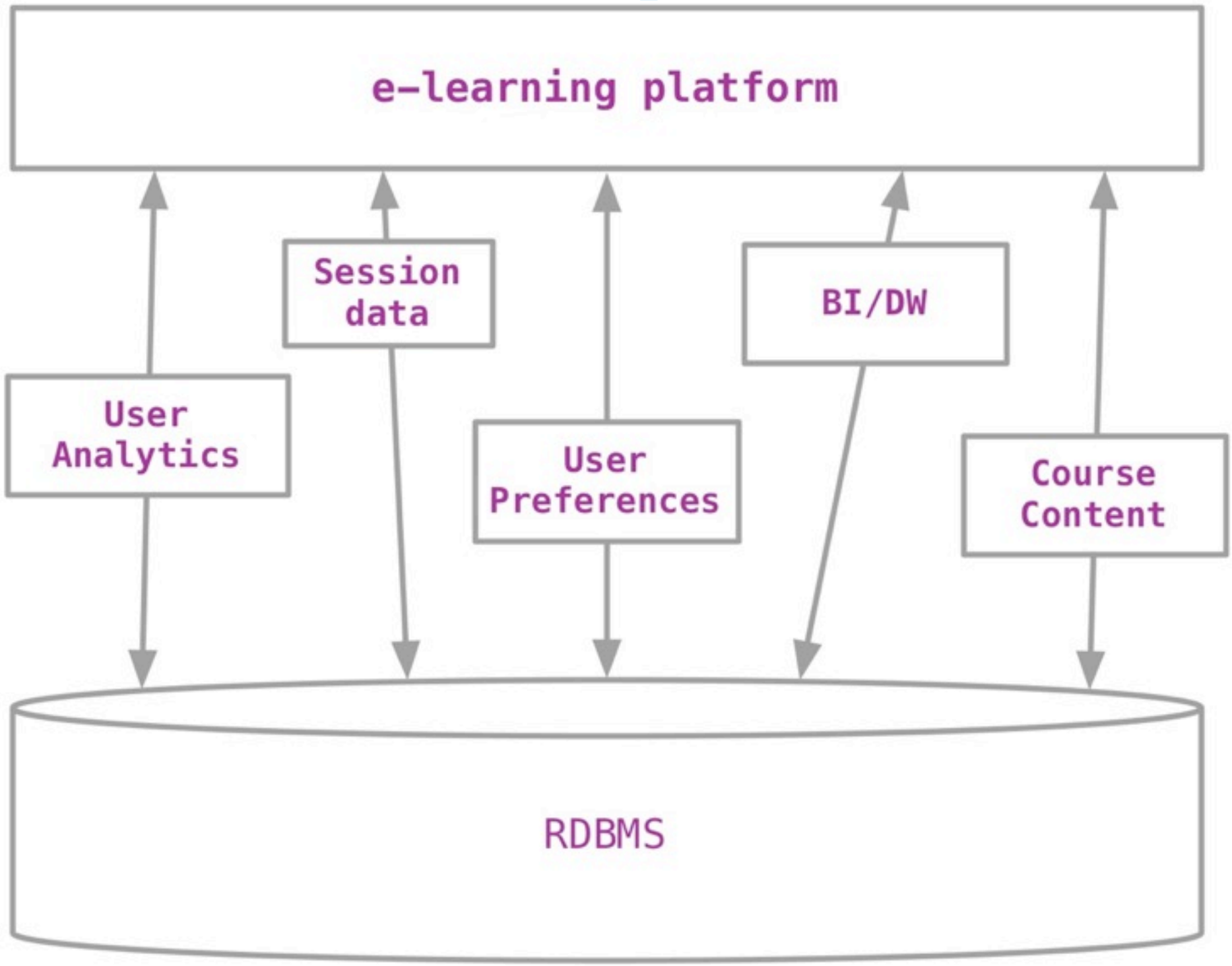
Session Storage	Financial Data	Shopping Cart	Recommendation engine
Redis	RDBMS	Riak	Neo4J

Product Catalog	Reporting	Analytics	User Activity Logs
MongoDB	RDBMS	Cassandra	Cassandra

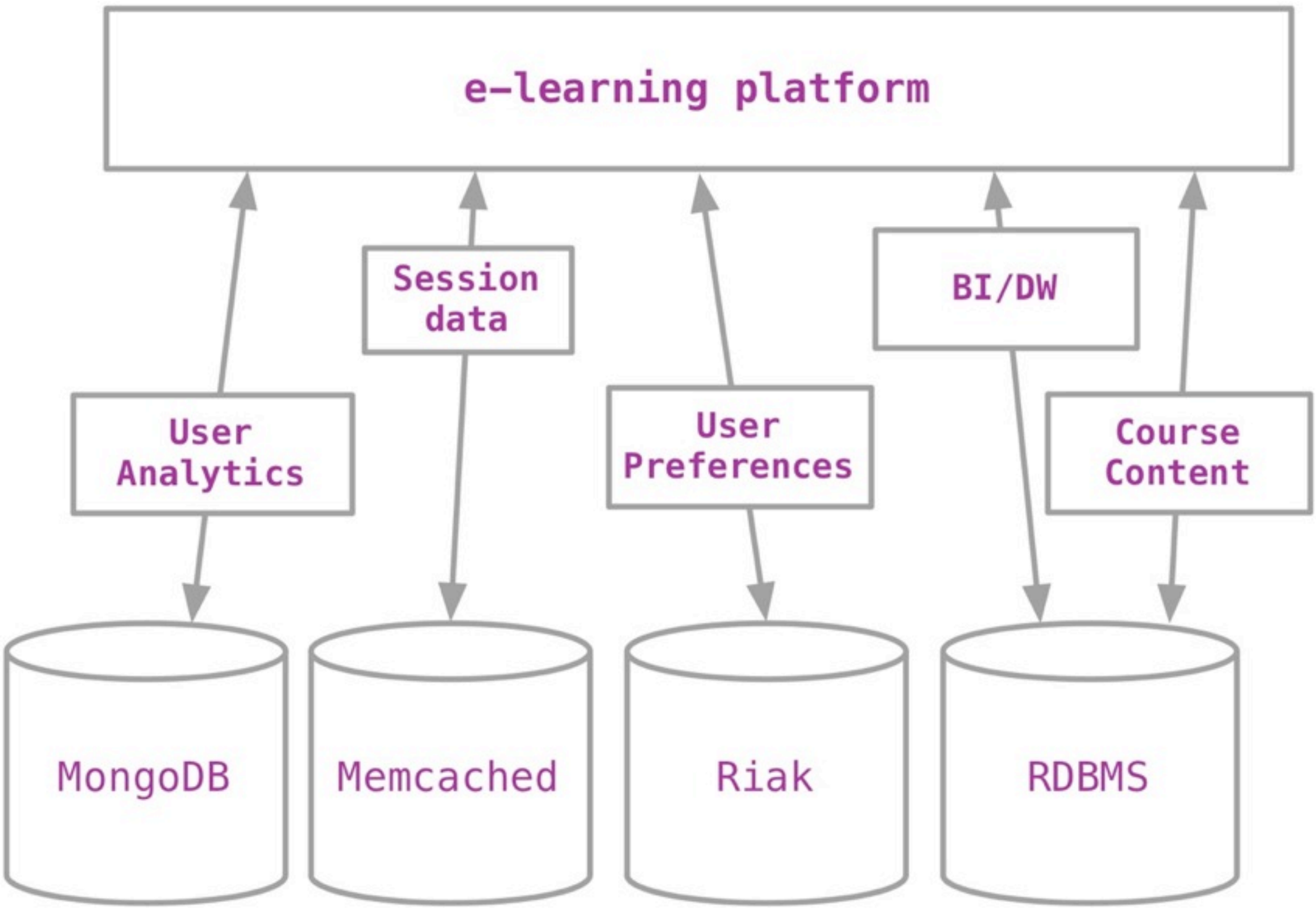
martinfowler.com/bliki/PolyglotPersistence.html

Experience

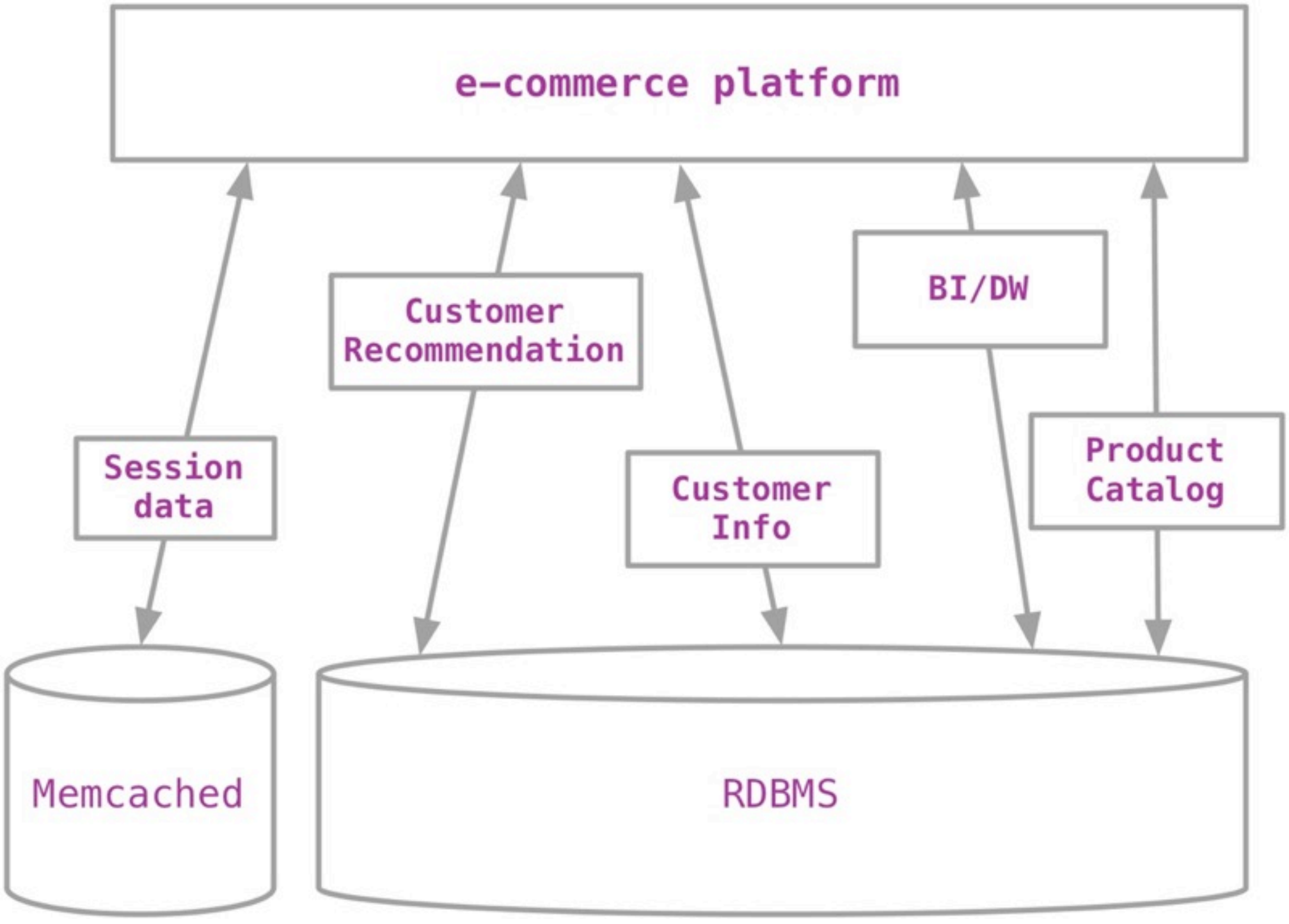
e-learning (before)



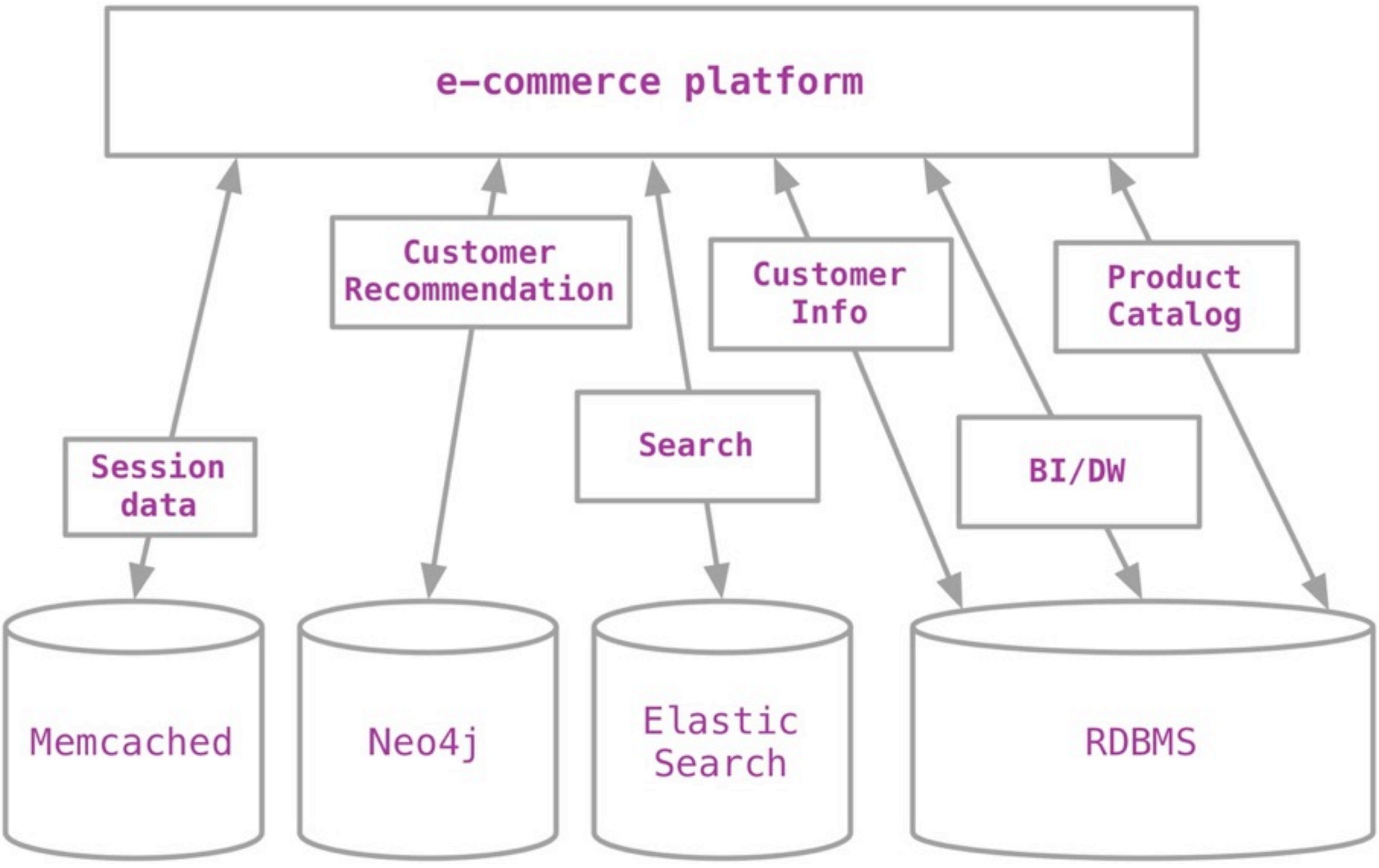
e-learning (after)



e-commerce (before)



e-commerce (after)



How do I choose?

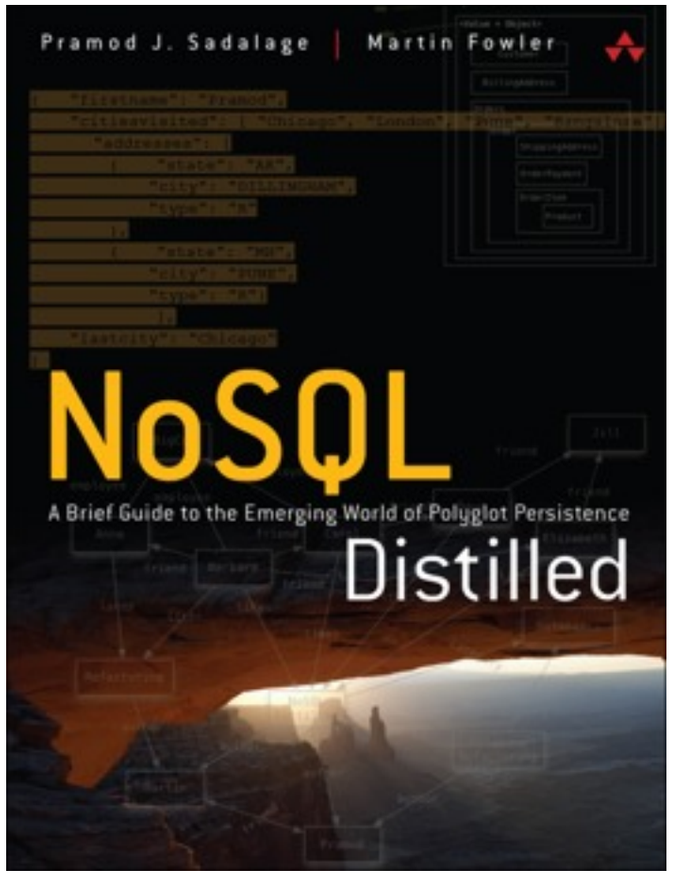
Choose for programmer productivity

Choose for data access performance

Choose to stick with the default

Choose by testing your expectations

Try the databases, they are all open-source



Thanks

#NoSQLDistilled
@pramodsadalage
sadalage.com