Couchbase

Introduction

Henry R.P

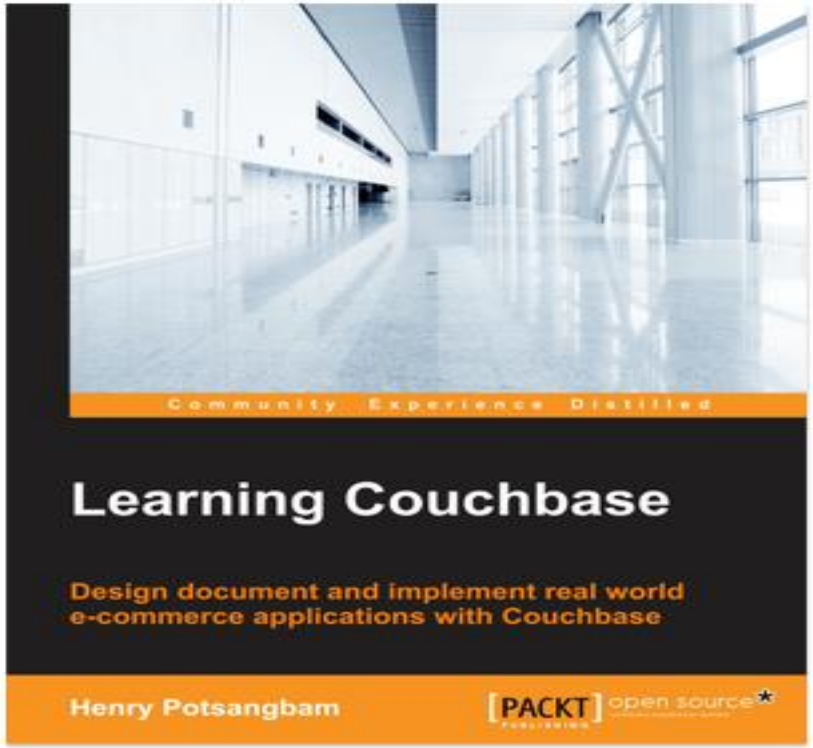
- **Mapr Certified Hadoop Administrator**
- **IBM Certified Application Developer**
- **IBM Certified Solution Designer**
- **SAP Certified Development Consultant.**

IT Architect , Author & Corporate trainer
15 +Year of IT Experience

- **TOGAF – Enterprise Architect**
- **CIPM – Certificate in Project Management.**

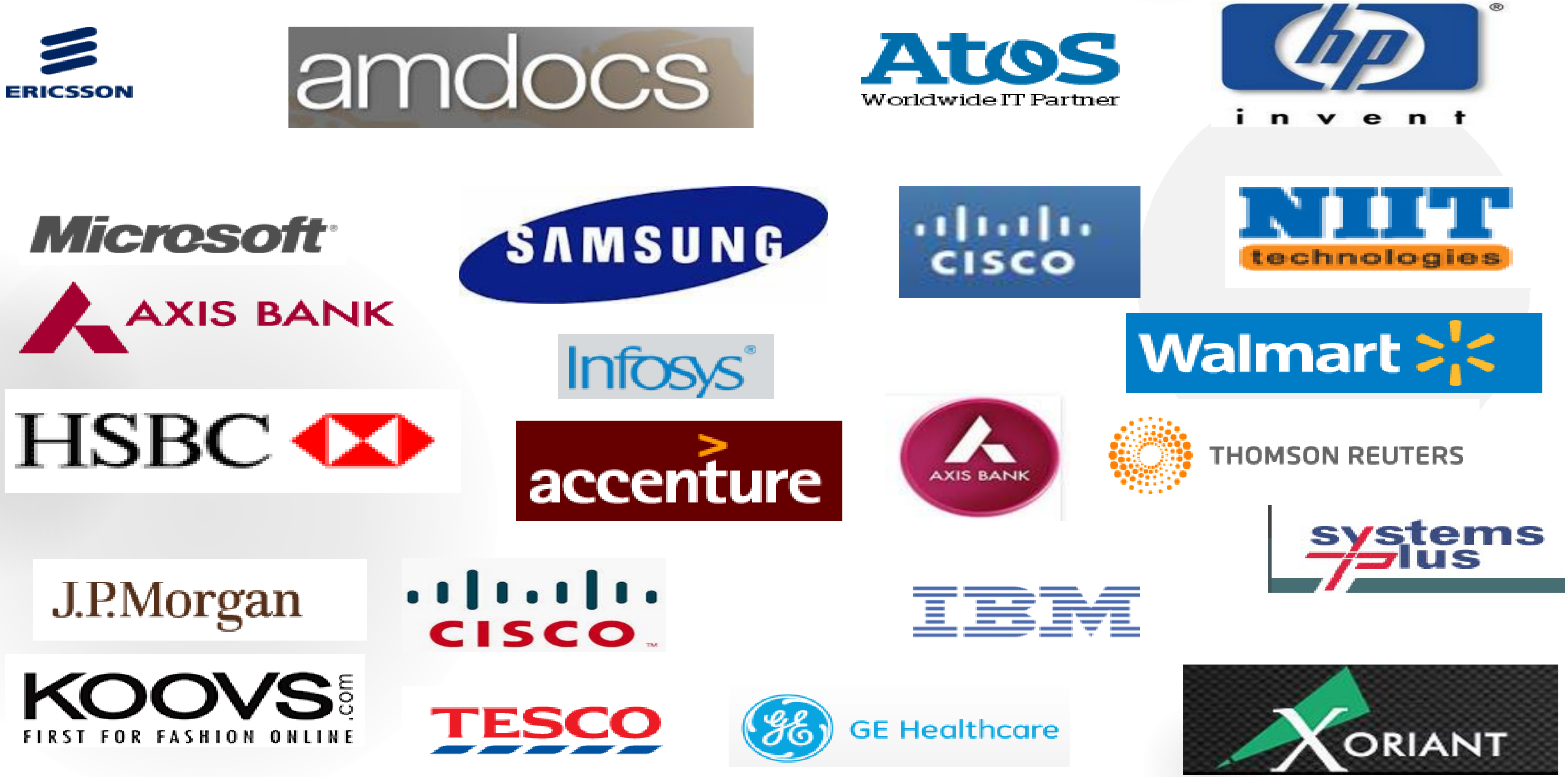
Training & Consulting on:

NOSQL & Bigdata – Hadoop, Couchbase, Cassandra, MongoDB,CDH, BigInsight
Predictive Analytics – R & SAS
EAI:- Mule / Fuse ESB /Spring Integration/ JBI / Apache Camel /Talend/ Apache Service Mix
Portal:- Liferay, SAP Netweaver.
Application server:- WAS, Tomcat , WebLogic, Jboss
Architecture: EA, TOGAF, CoBIT etc.
JEE Framework
OSGI - Eclipse PDE/Equinox/Virgo/Spring DM/ Felix / Karaf


Community Experience Distilled
Learning Couchbase
Design document and implement real world e-commerce applications with Couchbase
Henry Potsangbam
[PACKT] open source*

henry@thinkopensource.in

Clientele


ERICSSON
amdocs
AtoS
Worldwide IT Partner
hp
invent
Microsoft
SAMSUNG
CISCO
NIIT
technologies
AXIS BANK
Infosys
Walmart
HSBC
accenture
AXIS BANK
THOMSON REUTERS
systems plus
J.P.Morgan
CISCO
IBM
KOOVS.com
FIRST FOR FASHION ONLINE
TESCO
GE Healthcare
XORIANT

Introduce Yourself.

Name

Year of Experience.

Skills Level

- RDMS
- NoSql /Couchbase

Expectation, if any.

Outline

Overview - NoSQL Basic
Couchbase Server Architecture
Couchbase Administration - Webconsole
Bucket
Document Database Basic
Upgrade
Couchbse SDK - Overview
Cluster Administration
Views
N1QL
Security - LDAP
Client API – Java API , Rx and Spring DB
XDCR
FTS
Monitoring + Back up.

Time	
9.30 – 11.00 AM	Session I
11.00 AM to 11.15 AM	Tea Break
11.15 AM to 12.45 PM	Session II
12.45 PM to 1.45 PM	Lunch Break
1.45 PM to 3.15 PM	Session III
3.15 PM to 3.30 PM	Tea Break
3.30 PM to 5.30 PM	Session IV

An introduction to NoSQL databases

Relational databases

Benefits of Relational databases:

- Designed for all purposes
- ACID
- Strong consistancy, concurrency, recovery
- Mathematical background
- Standard Query language (SQL)
- Lots of tools to use with i.e: Reporting services, entity frameworks, ...
- Vertical scaling (up scaling)

Object / Object-relational databases were not practical. Mainly because of Impedance mismatch

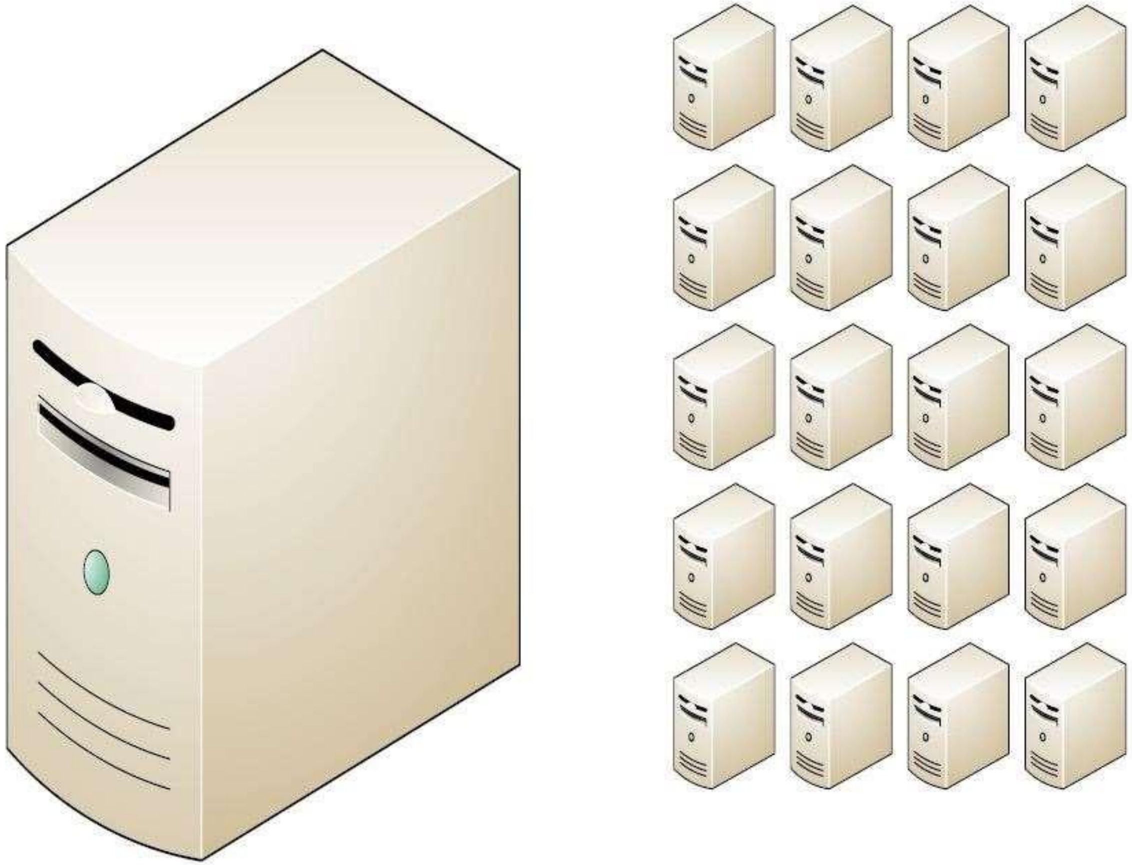
Era of Distributed Computing

But...

❑ Relational databases were not built for **distributed applications.**

Because...

❑ Joins are expensive
❑ Hard to scale horizontally
❑ Impedance mismatch occurs
❑ Expensive (product cost, hardware, Maintenance)



Era of Distributed Computing

But...

- □ Relational databases were not built for **distributed applications.**

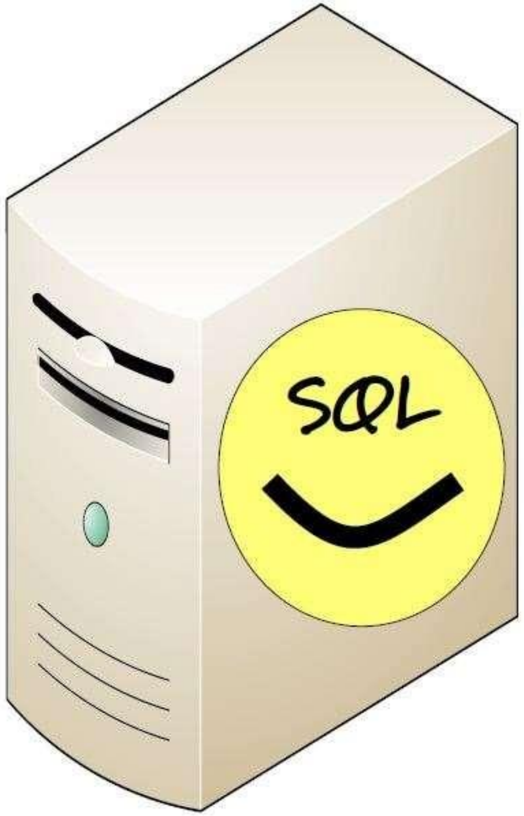
Because...

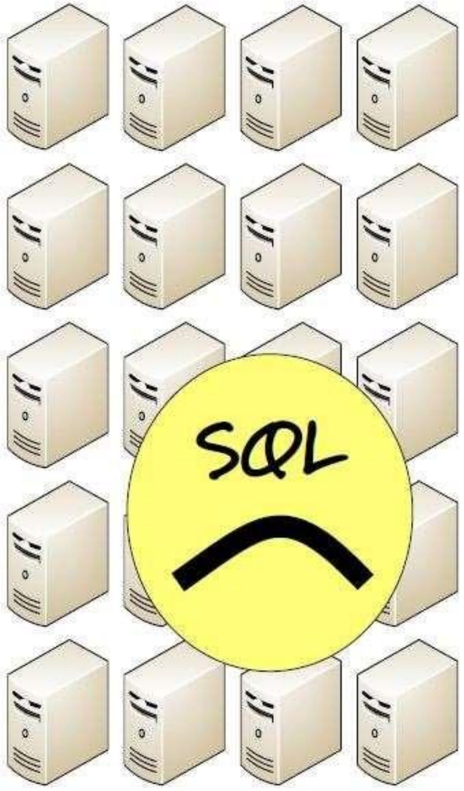
- □ Joins are expensive
- □ **Hard to scale horizontally**
- □ Impedance mismatch occurs
- □ Expensive (product cost, hardware, Maintenance)

And....

It's weak in:

- □ Speed (performance)
- □ High availability
- □ Partition tolerance





Characteristics of NoSQLdatabases

- Non relational
- Cluster friendly
- Schema-less
- 21 century web
- Open-source

Characteristics of NoSQL databases

NoSQL avoids:

- Overhead of ACID transactions
- Complexity of SQL query
- Burden of up-front schema design
- DBA presence
- Transactions (It should be handled at application layer)

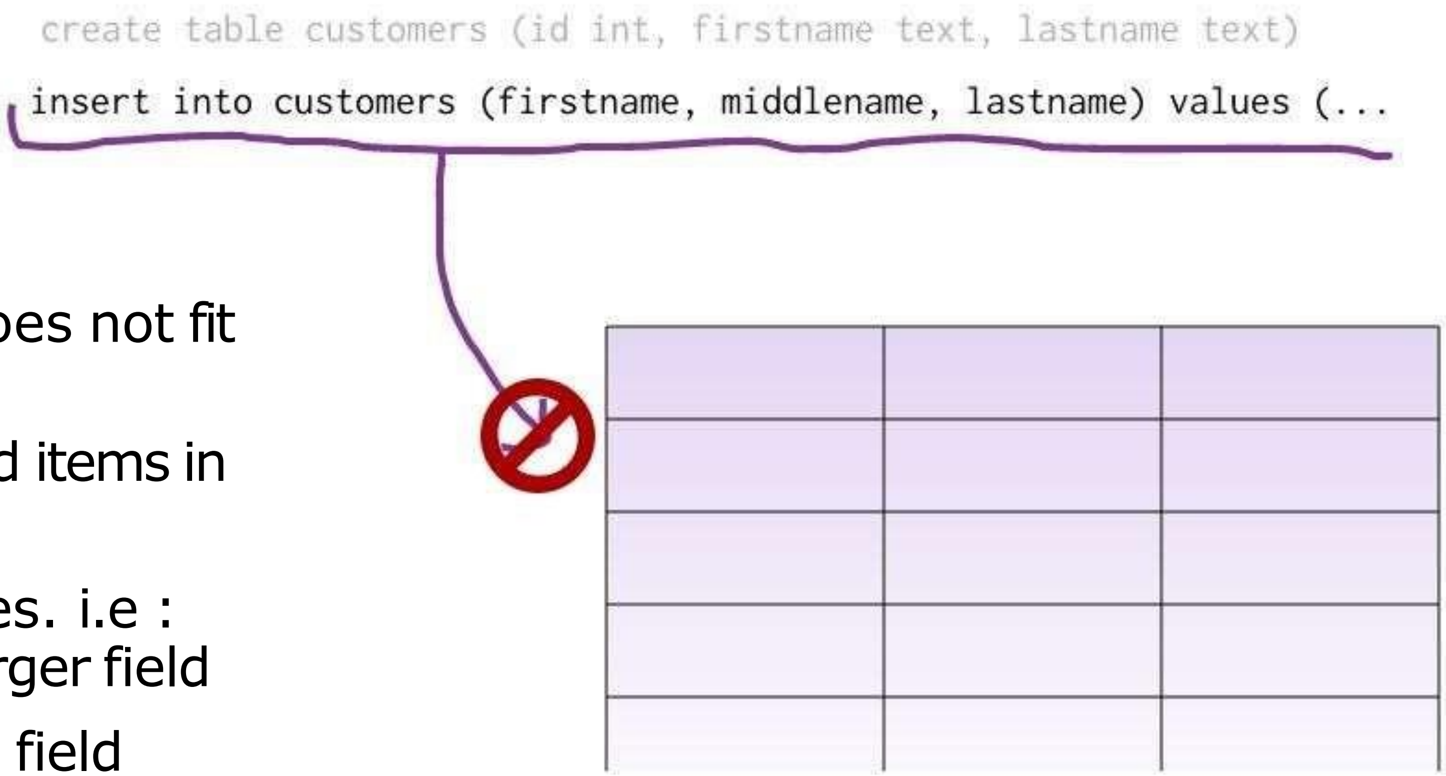
Provides:

- Easy and frequent changes to DB
- Horizontal scaling (scaling out)
- Solution to Impedance mismatch
- Fast development

What is a schema-less data model?

In relational Databases:

- You can’t add a record which does not fit the schema
- You need to add NULLs to unused items in a row
- We should consider the datatypes. i.e : you can’t add a stirng to an interger field
- You can’t add multiple items in a field (You should create another table: primary-key, foreign key, joins, normalization, ... !!!)

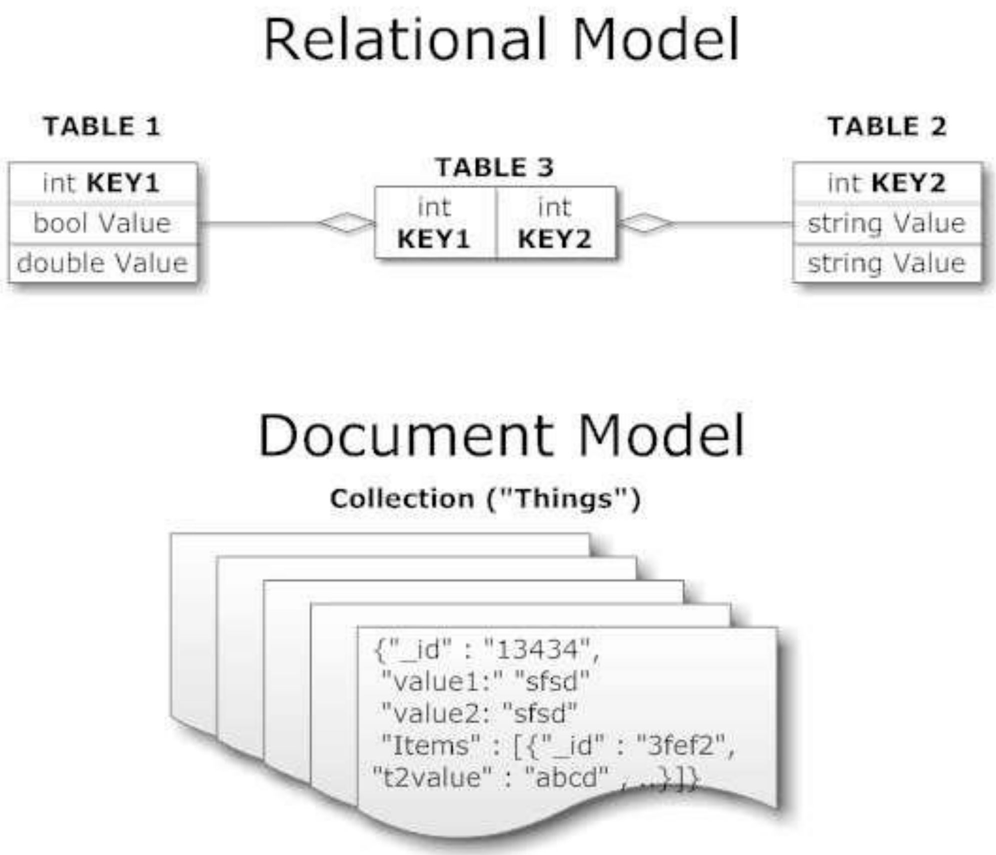


```
create table customers (id int, firstname text, lastname text)
insert into customers (firstname, middlename, lastname) values (...
```

What is a schema-less datamodel?

In NoSQL Databases:

- There is no schema to consider
- There is no unused cell
- There is no datatype (implicit)
- Most of considerations are done in application layer
- We gather all items in an aggregate (document)

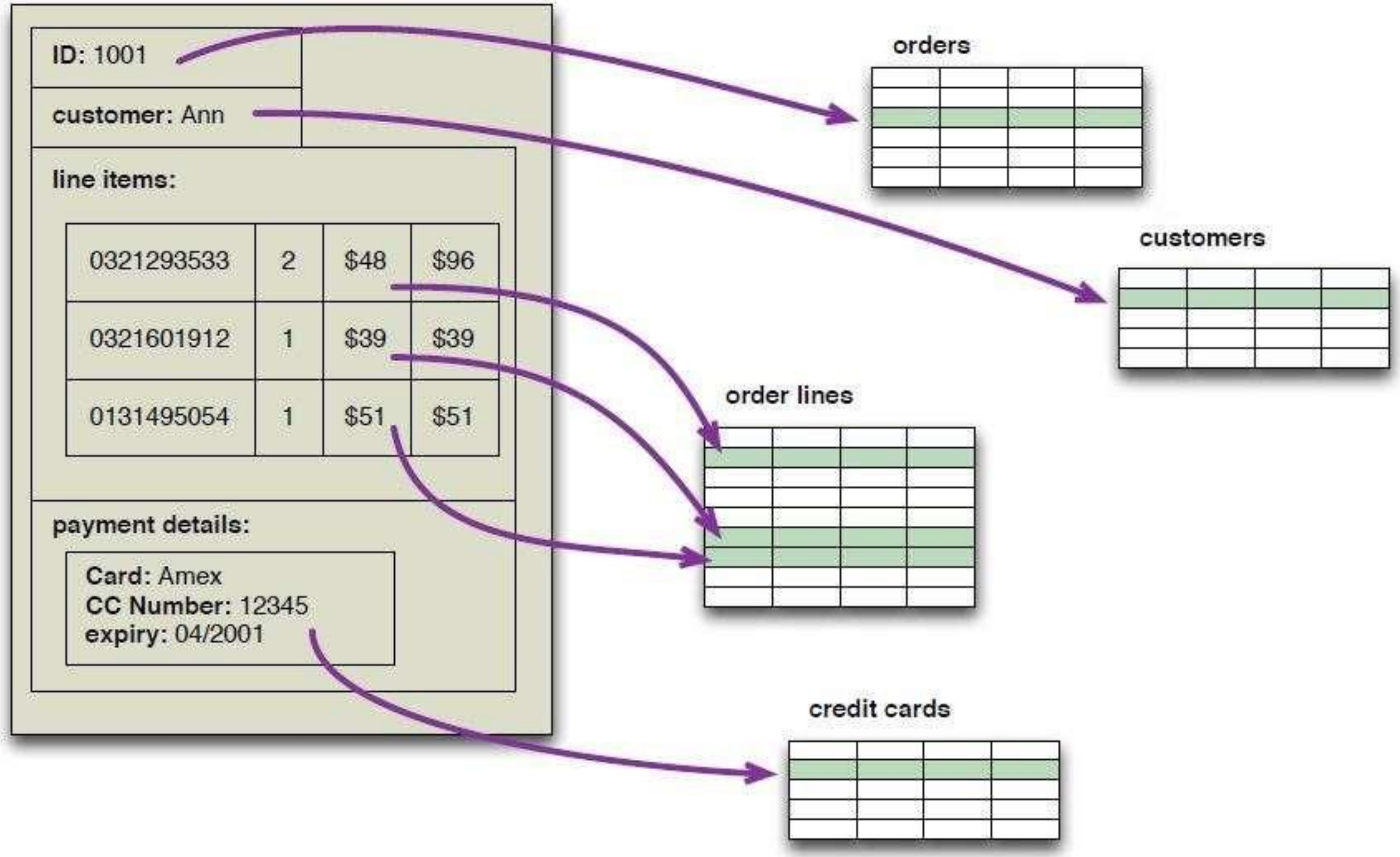


```
{"_id" : "13434",
 "value1:" "sfsd"
 "value2: "sfsd"
 "Items" : [{"_id" : "3fef2",
"t2value" : "abcd" , ...}]}
```

What is Aggregation?

- The term comes from Domain Driven Design
- Shared nothing architecture
- An aggregate is a cluster of domain objects that can be treated as a single unit
- Aggregates are the basic element of transfer of data storage - you request to load or save whole aggregates
- Transactions should not cross aggregate boundaries
- This mechanism reduces the join operations to a minimal level

What is Aggregation?



ID: 1001

customer: Ann

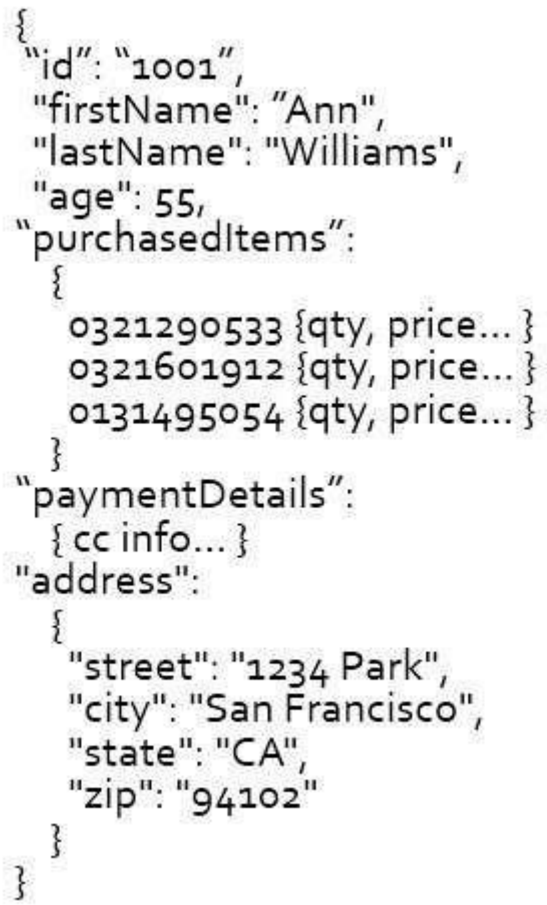
line items:

0321293533	2	$48	$96
0321601912	1	$39	$39
0131495054	1	$51	$51

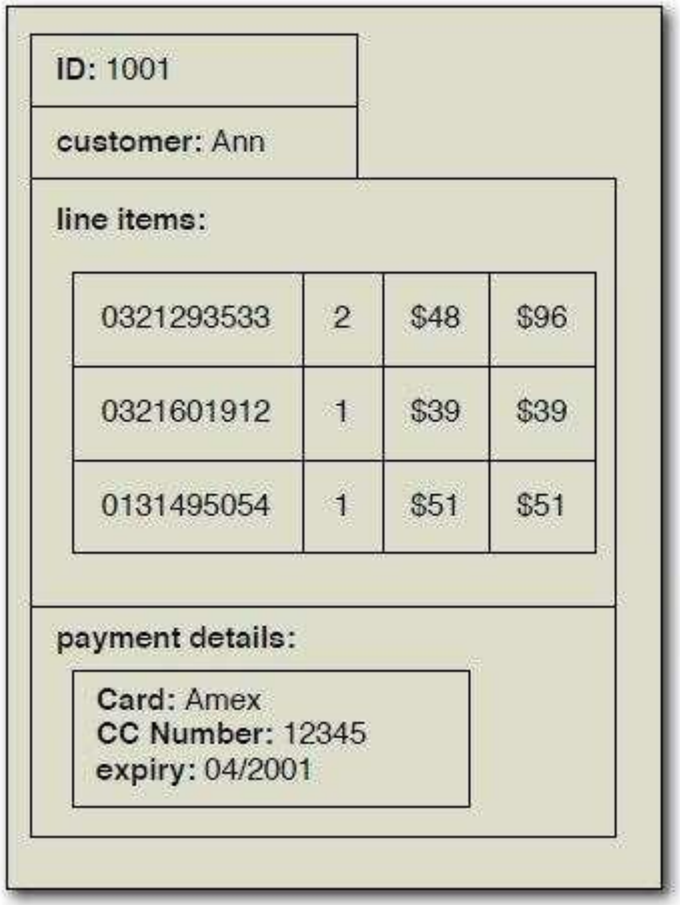
payment details:

Card: Amex
CC Number: 12345
expiry: 04/2001

What is Aggregation?



```
{
 "id": "1001",
  "firstName": "Ann",
  "lastName": "Williams",
  "age": 55,
"purchasedItems":
   {
     0321290533 {qty, price… }
     0321601912 {qty, price… }
     0131495054 {qty, price… }
   }
"paymentDetails":
   { cc info… }
"address":
   {
     "street": "1234 Park",
     "city": "San Francisco",
     "state": "CA",
     "zip": "94102"
   }
}
```



ID: 1001

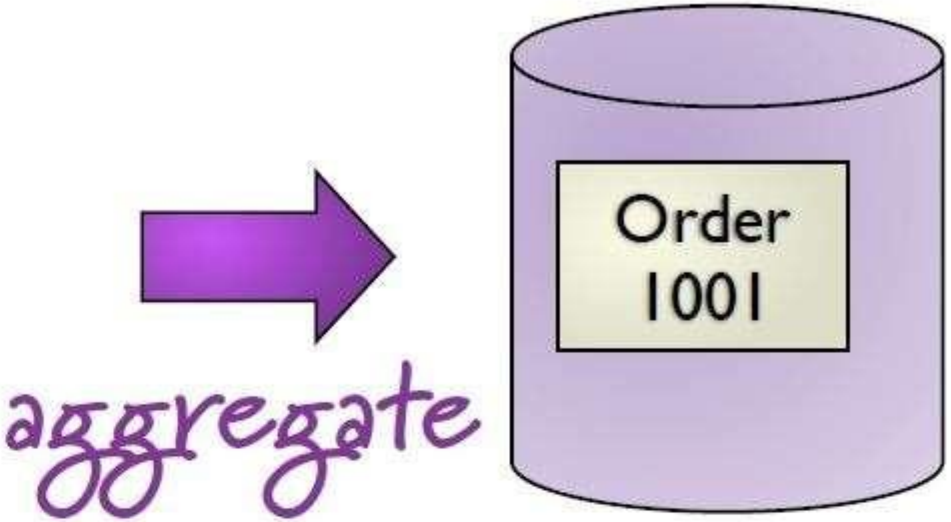
customer: Ann

line items:

0321293533	2	$48	$96
0321601912	1	$39	$39
0131495054	1	$51	$51

payment details:

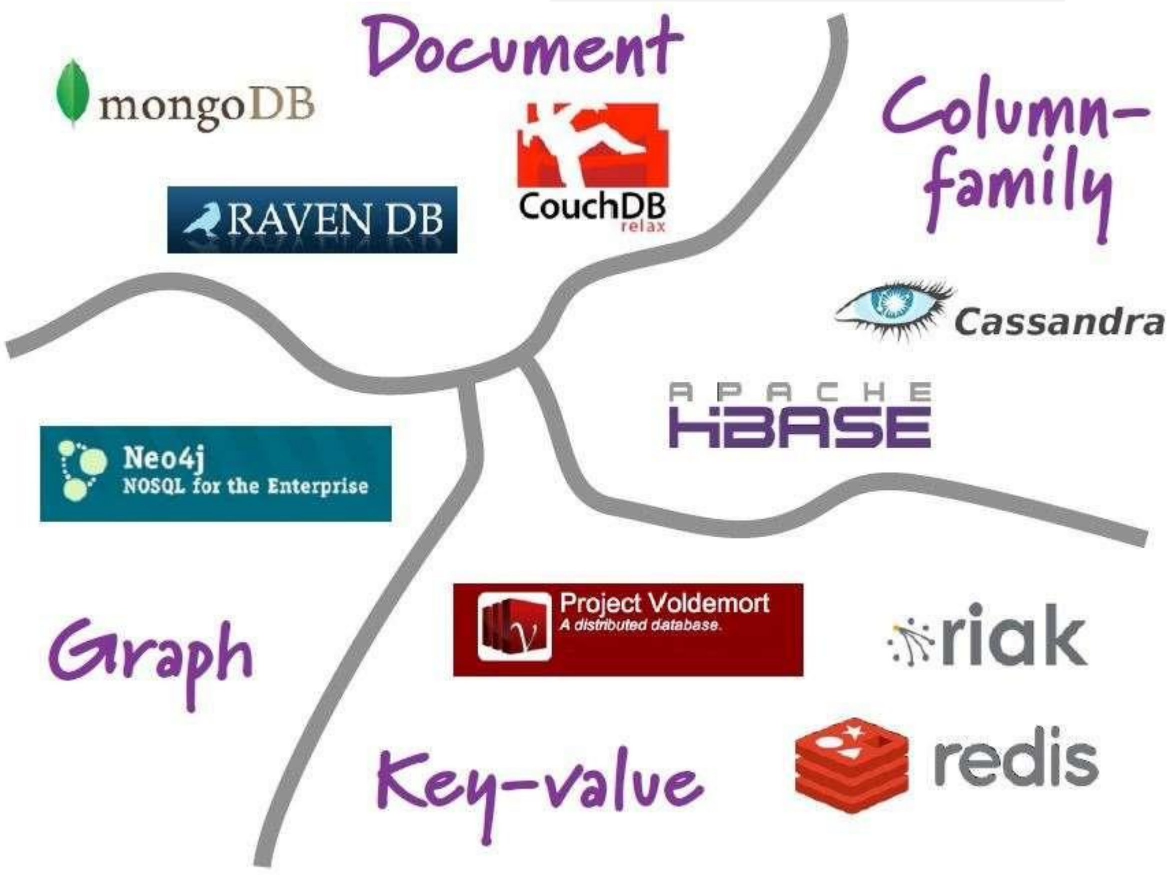
Card: Amex
CC Number: 12345
expiry: 04/2001



Aggregate Data Models

NoSQL databases are classified in four major datamodels:

- □ Key-value
- □ Document
- □ Column family
- □ Graph



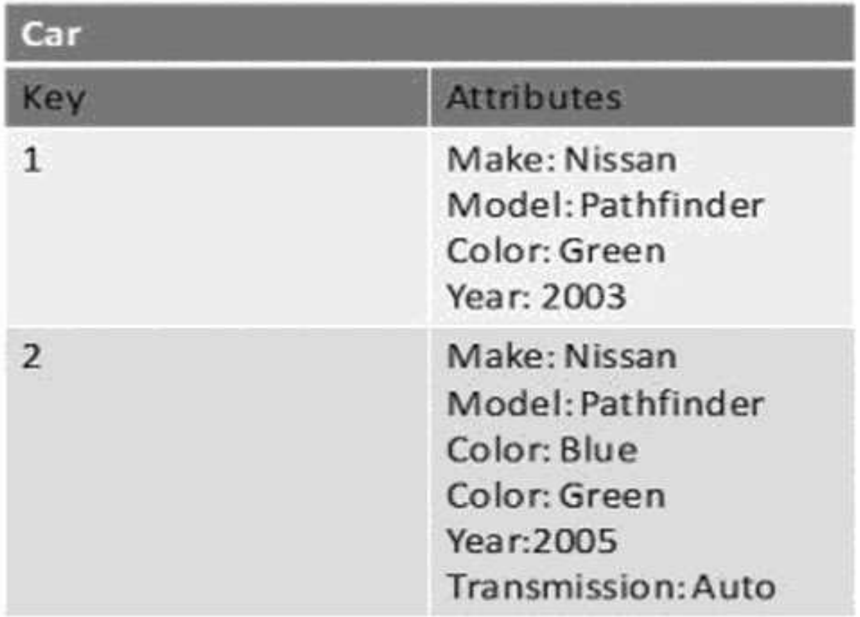
Each DB has its own query language

Key-value data model

- The main idea is the use of a hash table
- Access data (values) by strings called keys
- Data has no required format – data may have any format
- Data model: (key, value) pairs
- Basic Operations:

  Insert(key,value), Fetch(key),Update(key), Delete(key)

amazon DynamoDB

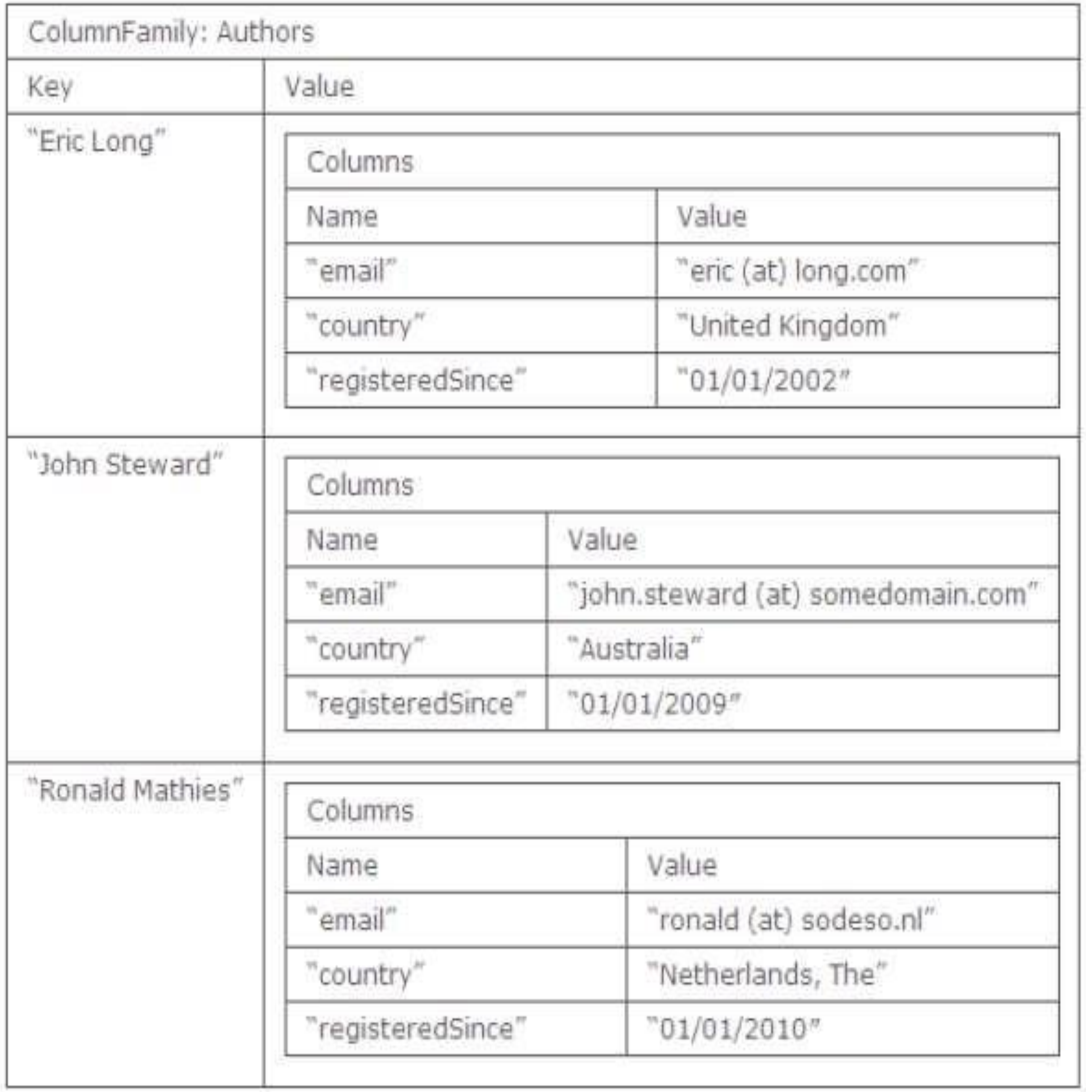


Car	
Key	Attributes
1	Make: Nissan Model: Pathfinder Color: Green Year: 2003
2	Make: Nissan Model: Pathfinder Color: Blue Color: Green Year:2005 Transmission: Auto

Column family data model

- **The column is lowest/smallest instance of data.**

  It is a tuple that contains a name, a value and a timestamp



ColumnFamily: Authors

Key	Value	
	Columns	
	Name	Value
"Eric Long"	"email"	"eric (at) long.com"
	"country"	"United Kingdom"
	"registeredSince"	"01/01/2002"
	Columns	
	Name	Value
"John Steward"	"email"	"john.steward (at) somedomain.com"
	"country"	"Australia"
	"registeredSince"	"01/01/2009"
	Columns	
	Name	Value
"Ronald Mathies"	"email"	"ronald (at) sodeso.nl"
	"country"	"Netherlands, The"
	"registeredSince"	"01/01/2010"

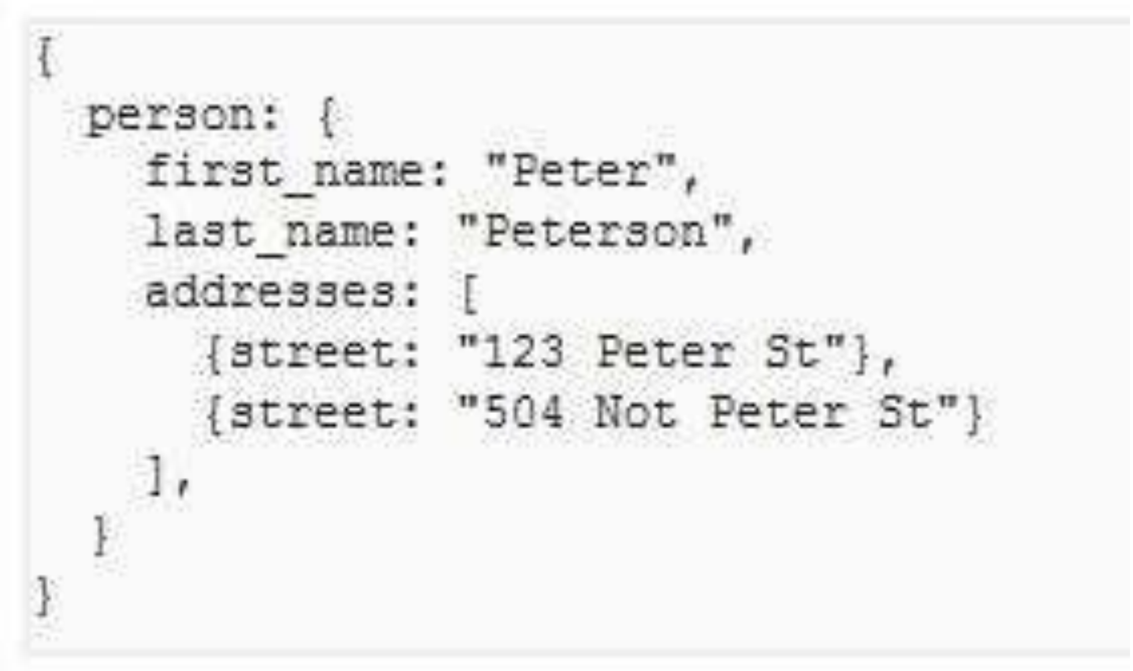
Graph data model

- Based on Graph Theory.
- Scale vertically, no clustering.
- You can use graph algorithms easily
- Transactions
- ACID

Neo4j
the world's
leading graph database

Document-based datamodel

- Usually JSON like interchange model.
- Query Model: JavaScript-like or custom.
- Aggregations: **Map/Reduce**
- Indexes are done via B-Trees.
- unlike simple key-value stores, both keys and values are fully searchable in document databases.



```
{
  person: {
    first_name: "Peter",
    last_name: "Peterson",
    addresses: [
      {street: "123 Peter St"},
      {street: "504 Not Peter St"}
    ],
  }
}
```

What we need?

- We need a distributed database system having such features:

- **–Fault tolerance**
- **–High availability**
- **–Consistency**
- **–Scalability**

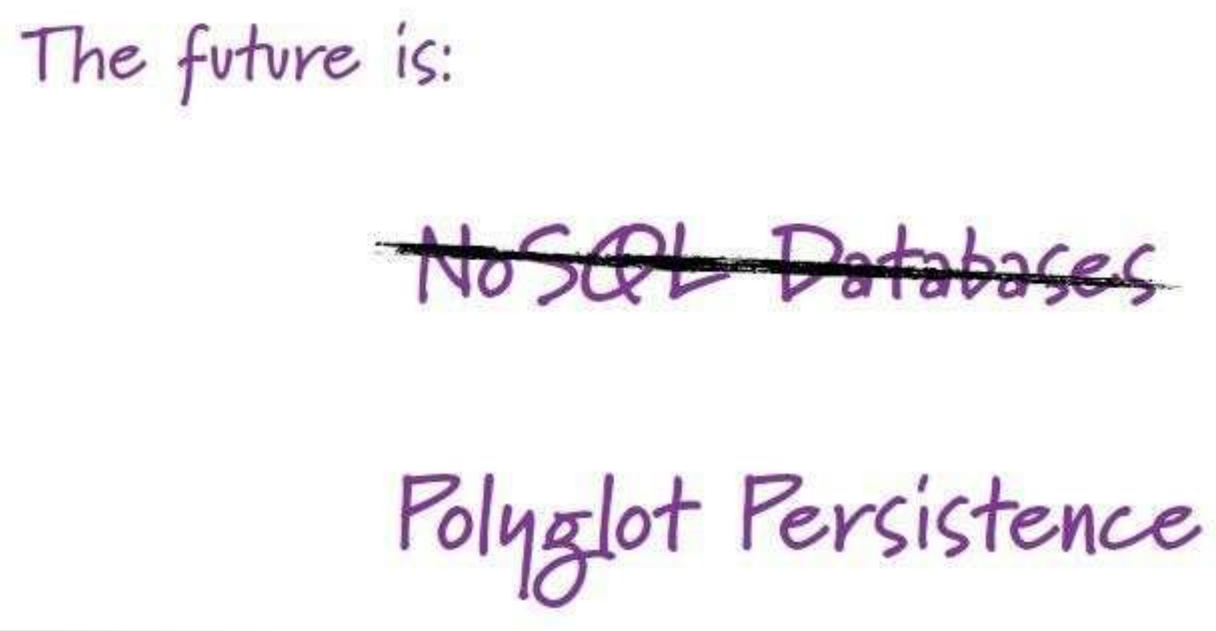
What we need?

- We need a distributed database system having such features:
  - **–Fault tolerance**
  - **–High availability**
  - **–Consistency**
  - **–Scalability**

Which is impossible!!!

According to CAP theorem

Polyglot persistence : the future of database systems



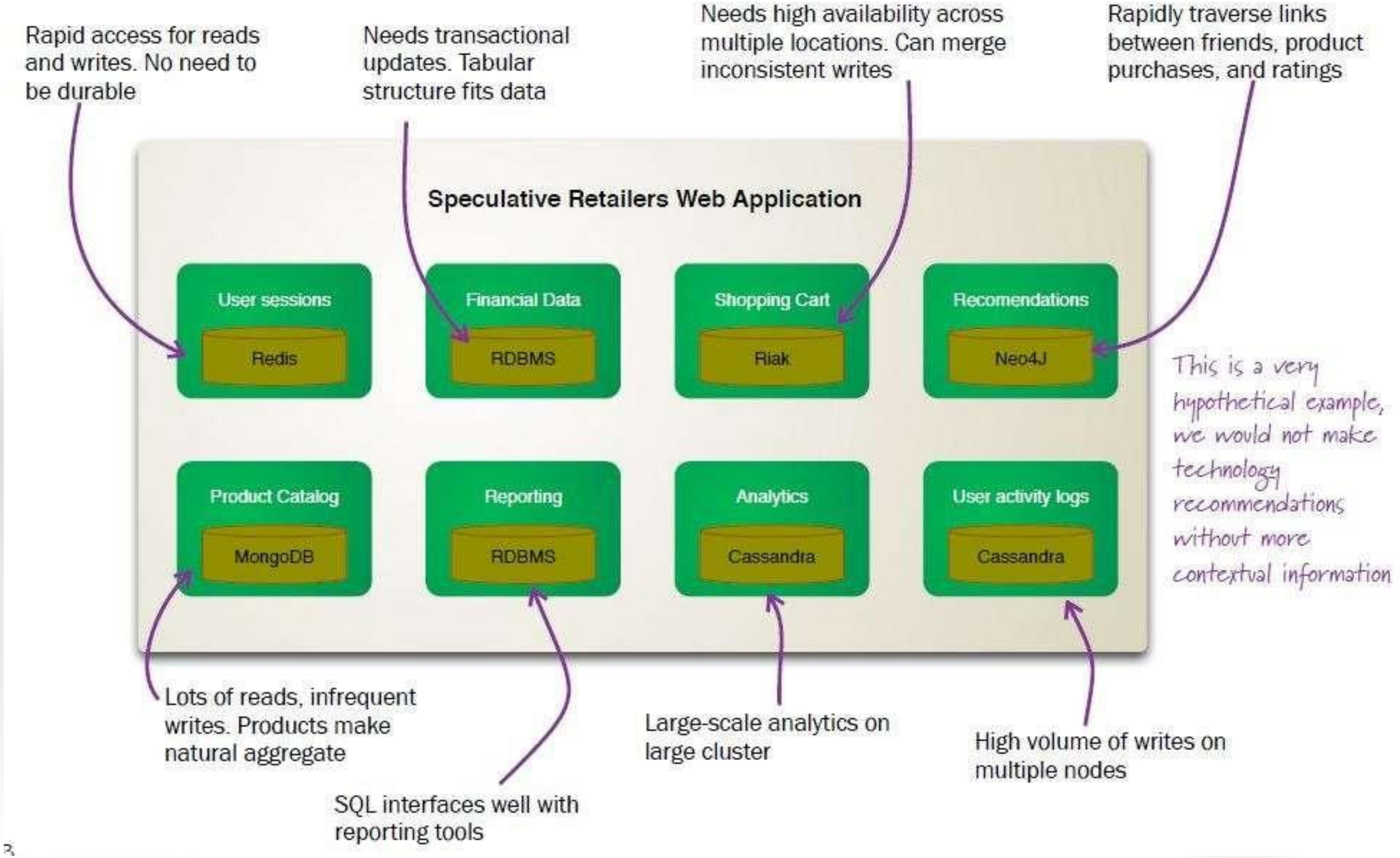
The future is:

~~NoSQL Databases~~

Polyglot Persistence

- Future databases are the combination of SQL & NoSQL
- We still need relational databases

Overview of a polygot db



Speculative Retailers Web Application

User sessions — Redis

Financial Data — RDBMS

Shopping Cart — Riak

Recomendations — Neo4J

Product Catalog — MongoDB

Reporting — RDBMS

Analytics — Cassandra

User activity logs — Cassandra

Rapid access for reads and writes. No need to be durable

Needs transactional updates. Tabular structure fits data

Needs high availability across multiple locations. Can merge inconsistent writes

Rapidly traverse links between friends, product purchases, and ratings

Lots of reads, infrequent writes. Products make natural aggregate

SQL interfaces well with reporting tools

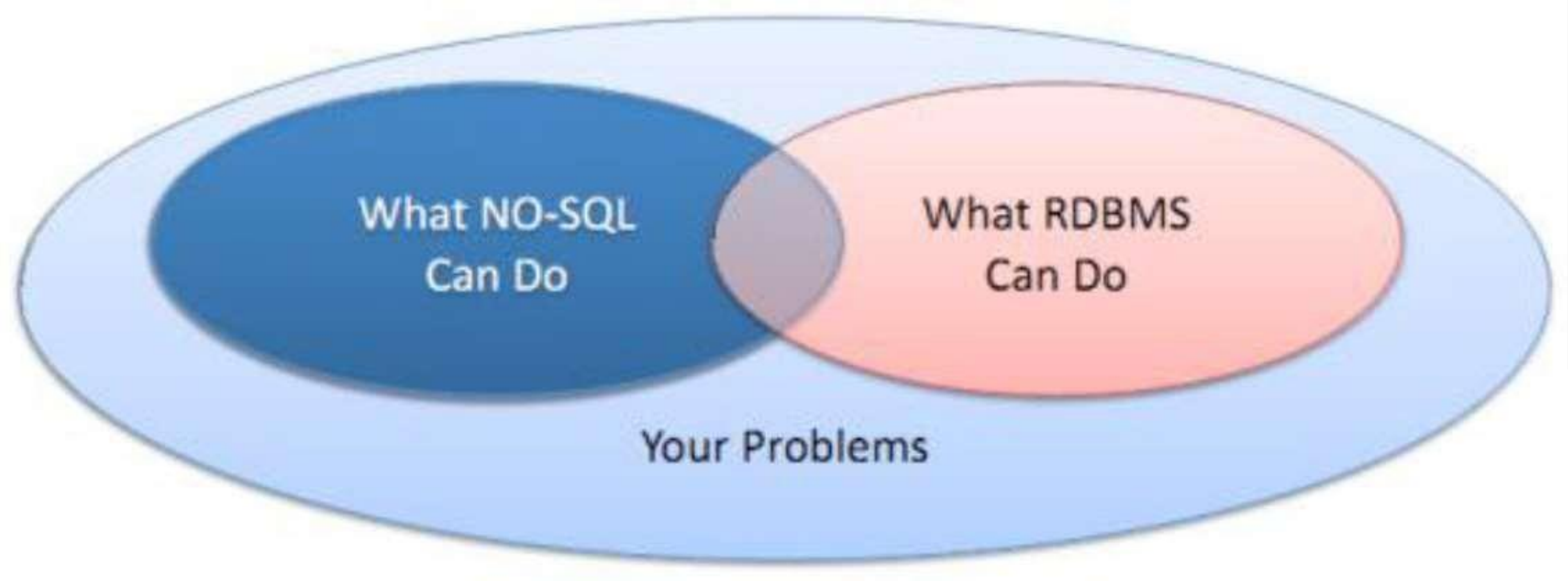
Large-scale analytics on large cluster

High volume of writes on multiple nodes

This is a very hypothetical example, we would not make technology recommendations without more contextual information

Conclusion:

Before you choose NoSQL as a solution:


What NO-SQL Can Do
What RDBMS Can Do
Your Problems