

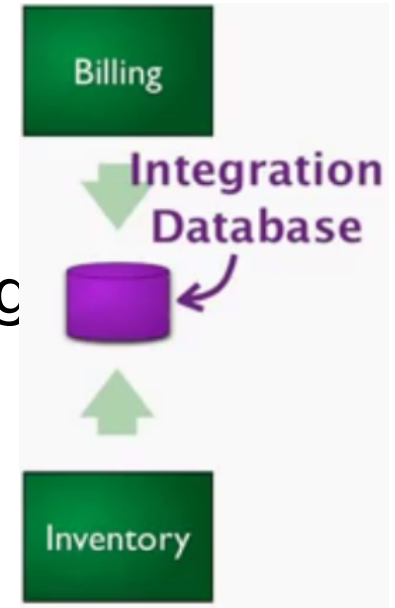
INTRODUCTION TO NOSQL DATABASES



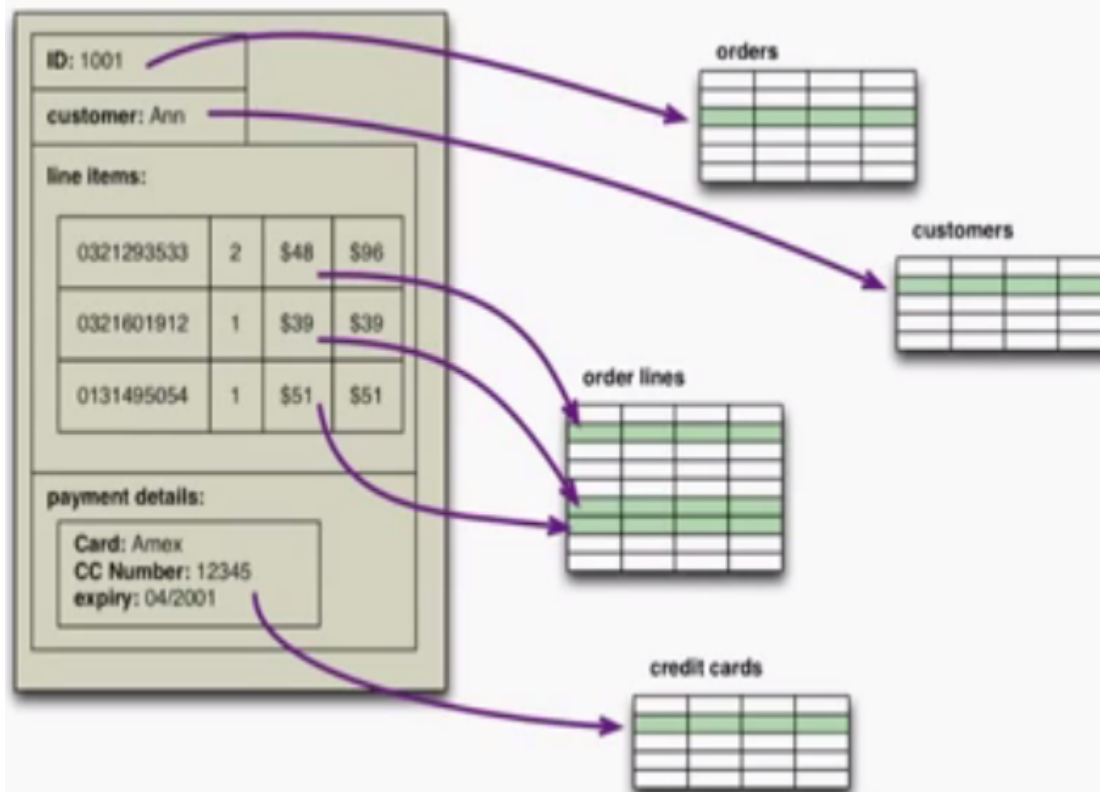
Background



- Rise of RDBMS Why
 - Persistence, Integration, SQL, Tx Mgt, Reporting
- Rise of OO Database
 - Don't really picked...
 - RDBMS still dominant technology
- Now What?



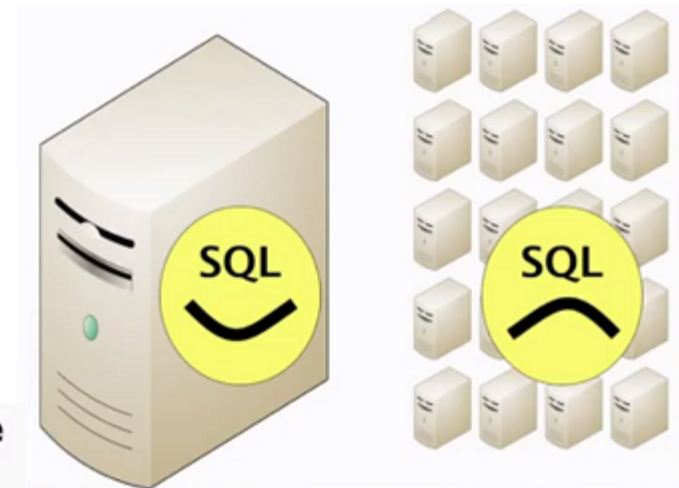
Problem with RDBMS?



- We have to manually map Requirement to different tables..
- It would be great if order can be stored as one Unit
- Impedance Mismatch=> ORM tools

Why No SQL

- Modern web application/ cloud create huge traffic
- Vertical Scale vs Horizontal scaling
- Horizontal scaling/ Hadoop is the solution=> RDBMS can not scale
- RDBMS is designed to run over big server rather than community H/W
- Running Relational DBMS over cluster is not easy!
- **Industry comes with new solutions**



Why NoSQL named NoSQL?

- John and big data guys propose meet up for alternative solution to RDBMS
- Need twitter hashtag #nosql

The screenshot shows a Meetup.com event page for 'NOSQL meetup'. The event is scheduled for Thursday, June 11, 2009, from 10:00 AM to 5:00 PM (PT) in San Francisco, CA. The 'Ticket Information' section shows a table with columns: TYPE, REMAINING, END, FREE, and QUANTITY. The row for 'Free ticket' shows 'Sold Out', 'Ended', 'Free', and 'Sold Out'. The 'When & Where' section includes a map of San Francisco with a red pin indicating the location. The 'Event Details' section has an 'Introduction' that states: 'This meetup is about "open source, distributed, non relational databases".'

NOSQL meetup
Thursday, June 11, 2009 from 10:00 AM to 5:00 PM (PT)
San Francisco, CA

TYPE	REMAINING	END	FREE	QUANTITY
Free ticket	Sold Out	Ended	Free	Sold Out

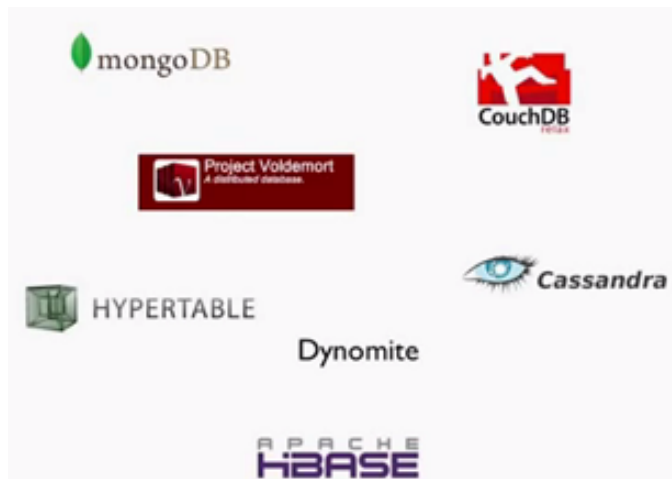
Share this! [Facebook](#) [Twitter](#) [LinkedIn](#) [Be the first of your friends to like this](#)

Event Details
Introduction
This meetup is about "open source, distributed, non relational databases".

When & Where
Map

Categories of NoSQL

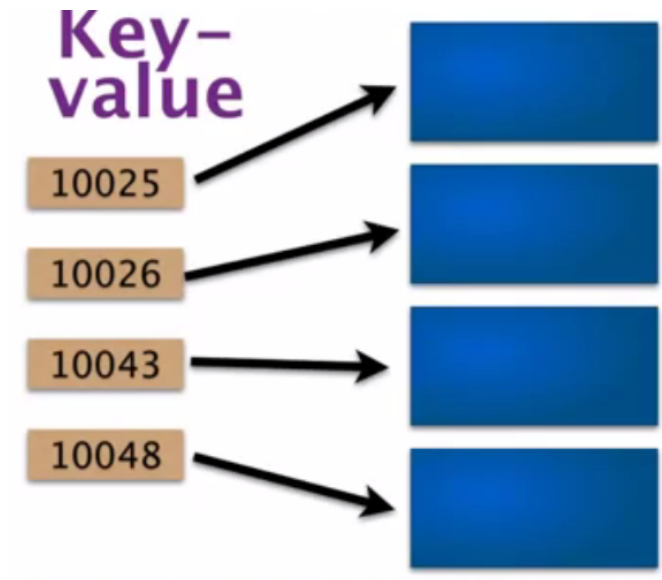
- Document
 - MongoDB, Couch DB
- Key value
 - Riak, Redis
- Column family
 - Cassandra, Hbase
- Graph
 - Neo4j



Characteristics of NoSQL

- Non Relational model
- Open Source
- Cluster friendly
- 21st century web
- Schema less design

Key-value store



- Simplest model
- Key have no idea what is actually stored against a key, it could be doc/image/Blob data
- Aka hashMap
- Table==Bucket

- Where should not be used?
 - Need relationship
 - Need Tx
 - ACID
 - Queries based on value eg: price>=200
- Use cases
 - Storing session information
 - User profile /Perference
 - Shopping cart

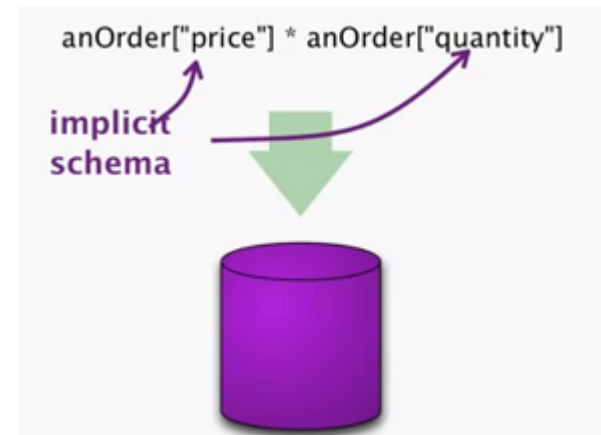
Document Data Model

Document

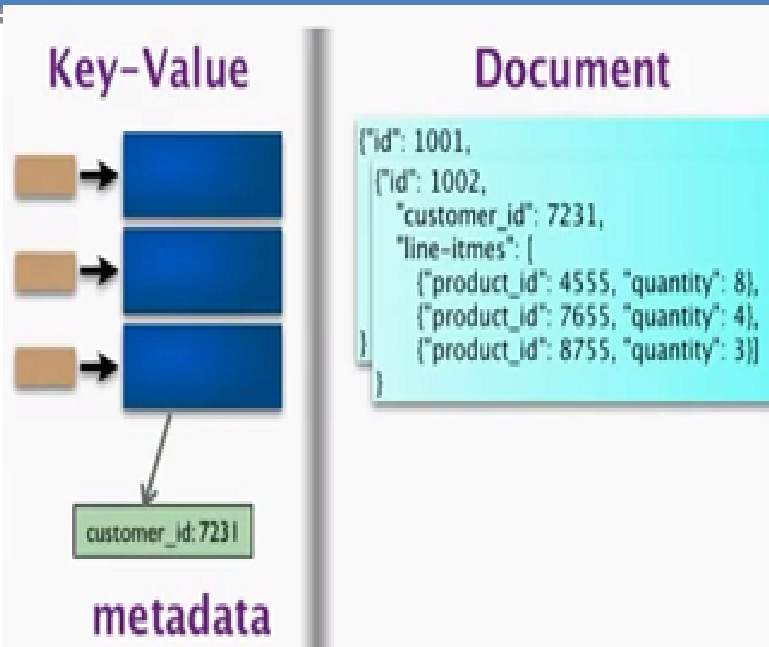
```
{ "id": 1001,  
  "customer_id": 7231,  
  "line-items": [  
    { "product_id": 4555, "quantity": 8 },  
    { "product_id": 7655, "quantity": 4 }, { "product_id": 8755,  
      "quantity": 2 } ],  
  "discount-code": "Y" }  
  
{ "id": 1002,  
  "customer_id": 9831,  
  "line-items": [  
    { "product_id": 4555, "quantity": 3 },  
    { "product_id": 2155, "quantity": 4 } ],  
  "discount-code": "Y" }
```

**no
schema**

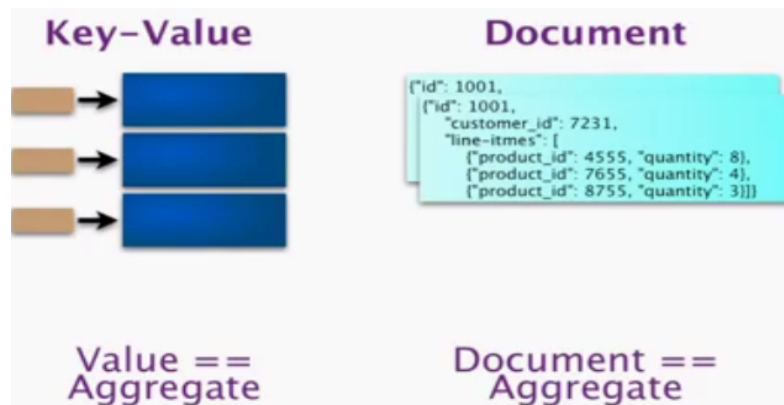
- Key -> Document (JSON)
- Flexible
- Can query to doc
- More transparent than key-value
- Provide implicit schema



Key value vs. Document



- Difference is very small
- Many key value database allowed use to have meta data like customer_id=7321
- Index is allowed



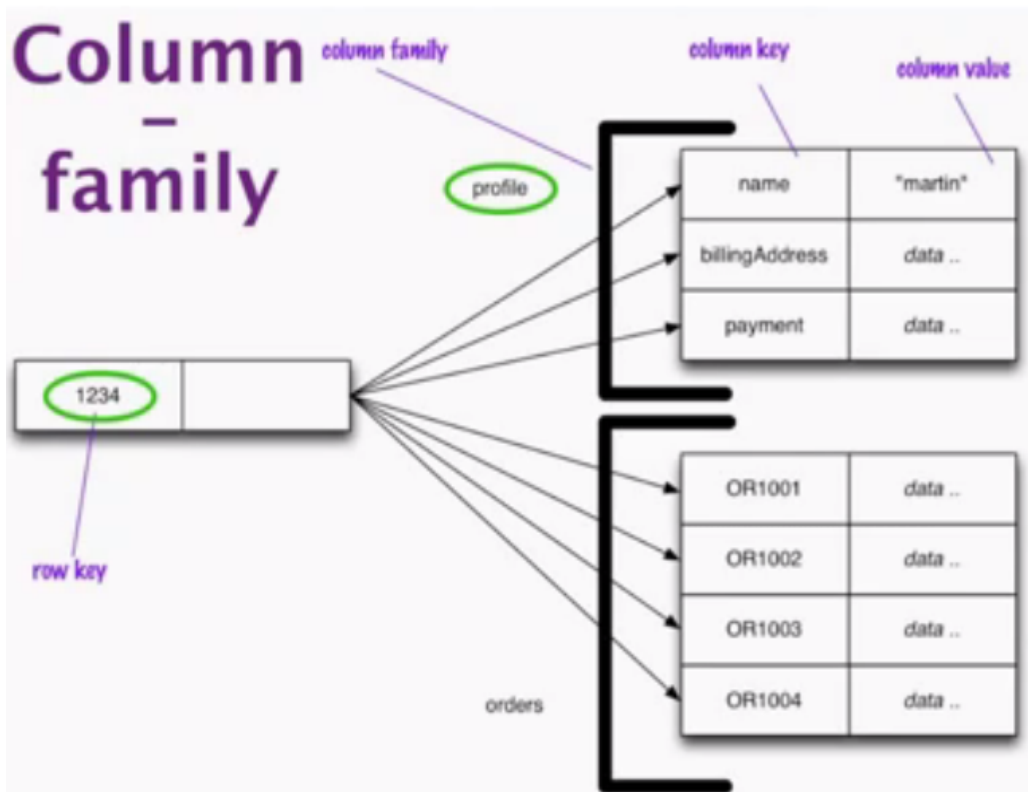
**Collectively called
Aggregate oriented
database**

Aggregate database



- Order is consider as a single unit that should be store in DB
- Very useful in case of distributed cluster: Each aggregate can be stored in an node without join etc, Only one node need to refer to access one order details.

Column family DB



- Aka Aggregate oriented DB
- More Complicated then earlier model
- Within one “Row key” we can store multiple column family, where each column family is combination of column that fit together
- Row Key+ column family name □ aggregate
- Easily retrieval of infidel elements

RDBMS vs Column db

<i>Employee_ID</i>	<i>Job</i>	<i>Dept</i>	<i>City</i>
1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

<i>Employee_ID</i>	<i>Job</i>	<i>Dept</i>	<i>City</i>
1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

Data stored in rows

1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

Data stored in columns

1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

INFOBRIGHT

<i>Employee_ID</i>	<i>Job</i>	<i>Dept</i>	<i>City</i>
1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston



Data stored in rows

1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston



INFOBRIGHT

Data stored in columns

1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

ID	job	dept	city
#			
#			
#			
#			
#			
#			

Row-Based Storage

ID	job	dept	city
#			
#			
#			
#			
#			
#			

Row Oriented works if...

- Transactional processing
- All the columns are required

Column-Based Storage

id	job	dept	city
#			
#			
#			
#			
#			
#			

Column Oriented works if...

- Analytical reporting
- Only relevant columns are required
- Reports are aggregates (sum, count, average, etc.)