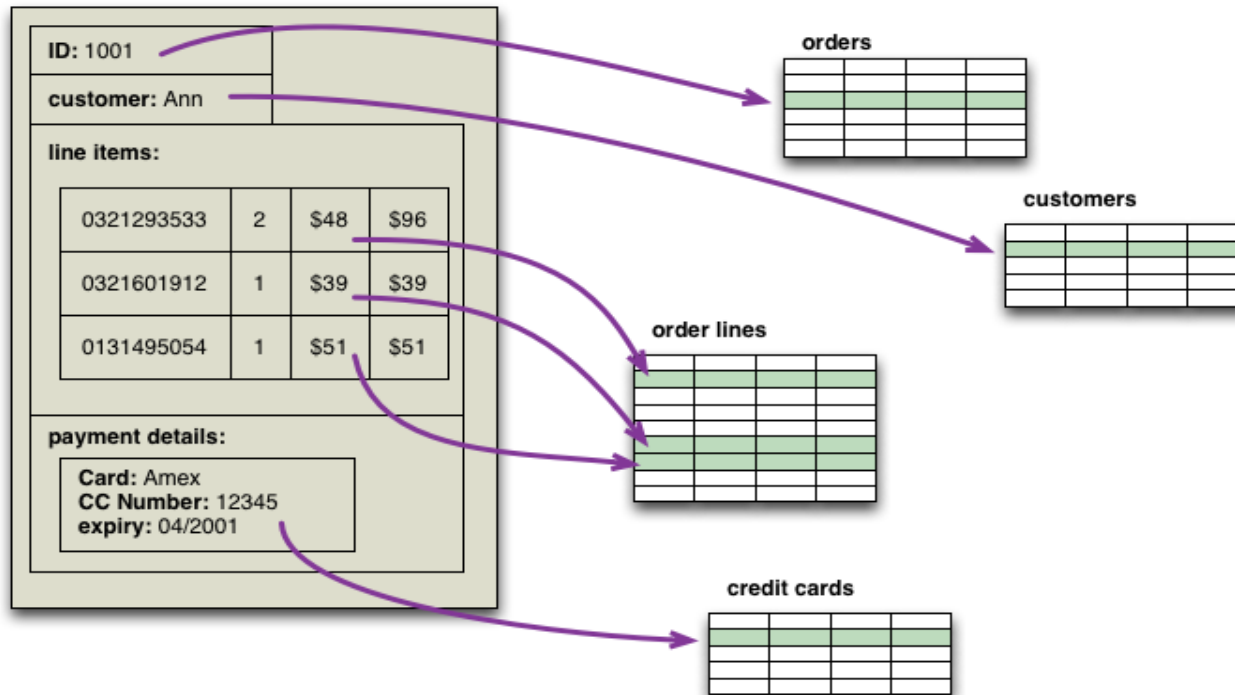


# Document Database Basics Couchbase

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address":
  {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber":
  [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ]
}
```

# The relational approach to model data



*Relational databases were not designed with clusters in mind, which is why people have cast around for an alternative. Storing aggregates as fundamental units makes a lot of sense for running on a cluster.*

<http://martinfowler.com/bliki/AggregateOrientedDatabase.html>

# Compared to a Document Database

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			



```
o::1001
{
  uid: "ji22jd",
  customer: "Ann",
  line_items: [
    { sku: 0321293533, quan: 3, unit_price: 48.0 },
    { sku: 0321601912, quan: 1, unit_price: 39.0 },
    { sku: 0131495054, quan: 1, unit_price: 51.0 }
  ],
  payment: {
    type: "Amex",
    expiry: "04/2001",
    last5: 12345
  }
}
```

- Easy to distribute data
- Makes sense to application programmers

# Object to JSON back to Object

User Object

string	uid
string	firstname
string	lastname
int	age
array	favorite_colors
string	email



u::michael.nitschinger@couchbase.com

```
{
  "uid": 123456,
  "firstname": "Michael",
  "lastname": "Nitschinger",
  "age": 24,
  "favorite_colors": ["blue", "black"],
  "email": "michael.nitschinger@cou..."
}
```



set()



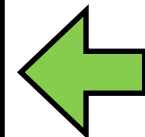
User Object

string	uid
string	firstname
string	lastname
int	age
array	favorite_colors
string	email



u::michael.nitschinger@couchbase.com

```
{
  "uid": 123456,
  "firstname": "Michael",
  "lastname": "Nitschinger",
  "age": 24,
  "favorite_colors": ["blue", "black"],
  "email": "michael.nitschinger@cou..."
}
```



get()



# Using a Document-Oriented Approach

- use documents in a specific format that stores data representing an object graph.
- documents are stored in JSON format.
- we do not represent the fields with null values.

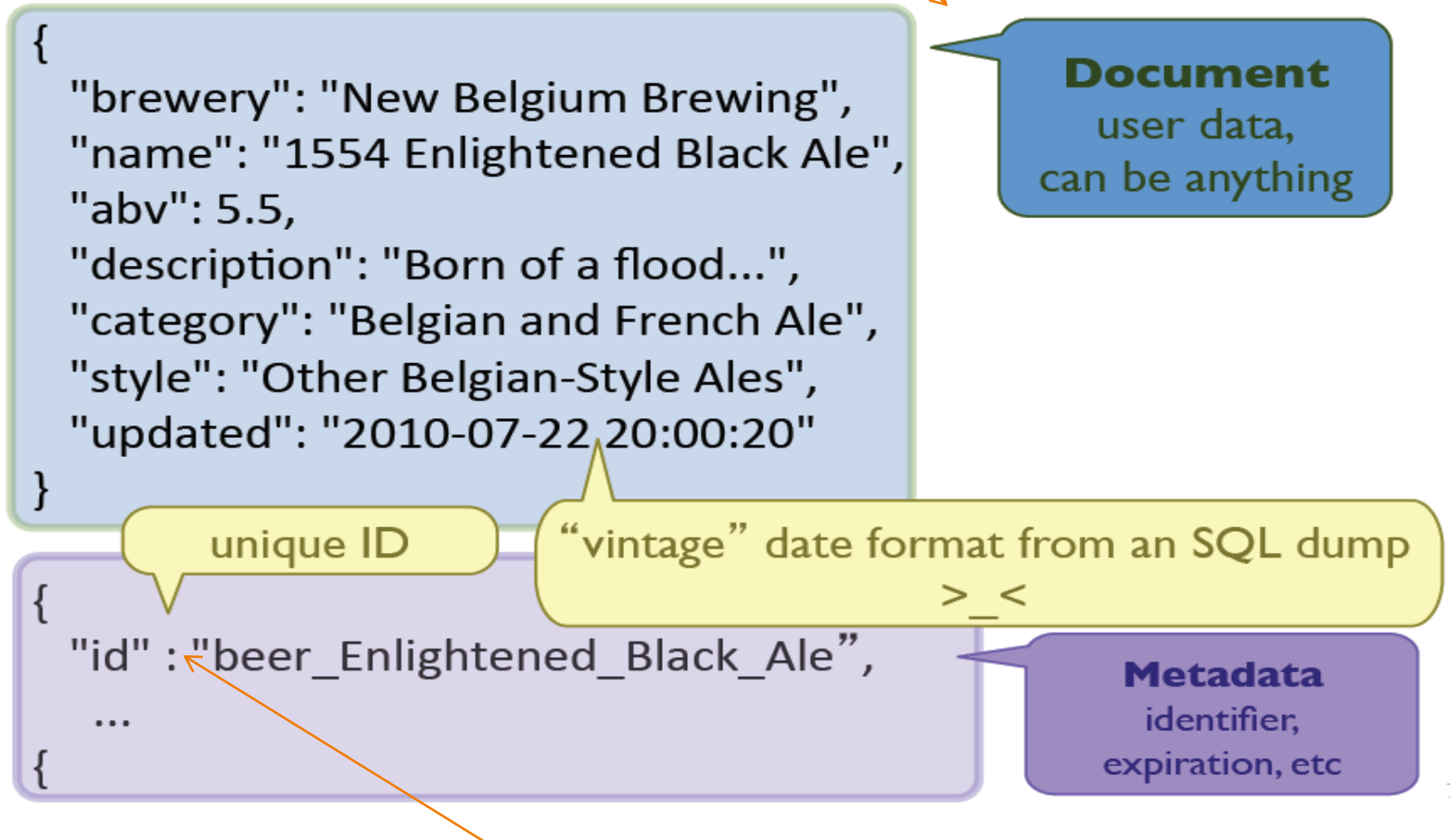
```
{  
  "id": "2d18e287-d5fa-42ce-bca5-b26eae0d7d4",  
  "type": "rant",  
  "userName": "JerryS",  
  "rantText": "Why do they call it Ovaltine? The mug is round. The jar is round. They should  
call it Roundtine."  
}
```



Not bound to a schema.

# Meta + Document Body

Most Recent In Ram And Persisted To Disk



All Keys Unique and Kept in RAM

## ■ Using Admin UI

name ▼	items	resident	ops/sec	RAM used/quota	disk used	
beer-sample	7,305	100%	0	16.8MB / 100MB	13.4MB	<a href="#">Documents</a> <a href="#">S</a>
couchmusic2	213,063	100%	0	258MB / 356MB	232MB	<a href="#">Documents</a> <a href="#">S</a>
default	1	100%	0	11.9MB / 256MB	4.08MB	<a href="#">Documents</a> <a href="#">S</a>
travel-sample	31,591	100%	0	55.8MB / 100MB	106MB	<a href="#">Documents</a> <a href="#">S</a>

ministrator > Buckets > Documents

**ADD DOCUMENT**

default ▼



filter: ?skip=0&include\_docs=true&limit=11

Document ID

Look Up ID

ID

content sample

life.skolur@gmail.com

{"name":"Henry P","password":"123456","userName":"Henryp","docType":"user","email":"life.skolur@gmail.com"}

Delete

Edit

10 ▼ per page

< page 1 >





- Document ID → Enter document detail

Create Document X

Document ID

LearningCB

Cancel Save Document

Enter the document details

LearningCB

Delete

Save As...

Save

```
1 {
2   "click": "to edit",
3   "with JSON": "there are no reserved field names"
4 }
```

```
1 {
2   "id": "LearningCB",
3   "rev": "1-
15154f78cb6a0000000000002000006",
4   "expiration": 0,
5   "flags": 33554438
6 }
```

- **Labs:**  
**Couchbase Web Console**  
**Change Bucket Settings**