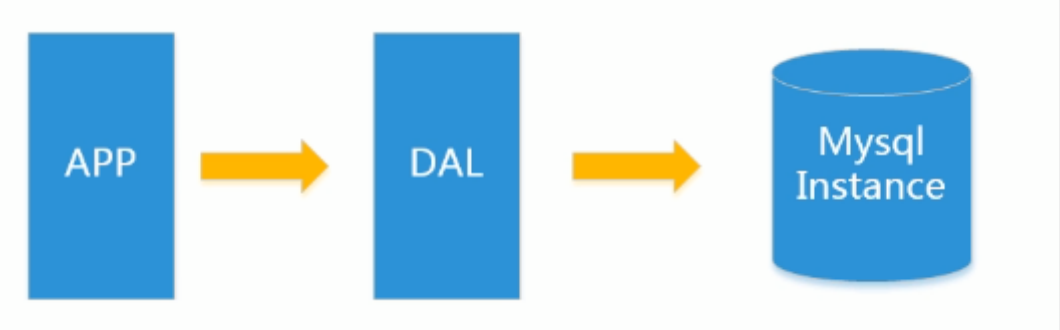
NoSql概述

1. 互联网时代背景下大机遇，为什么用NoSql

1.1 单机MySql的美好时代

在90年代，一个网站的访问量一般都不大，用单个数据库完全可以轻松应付。

在那个时候，更多的都是静态网页，动态交互类型的网站不多。

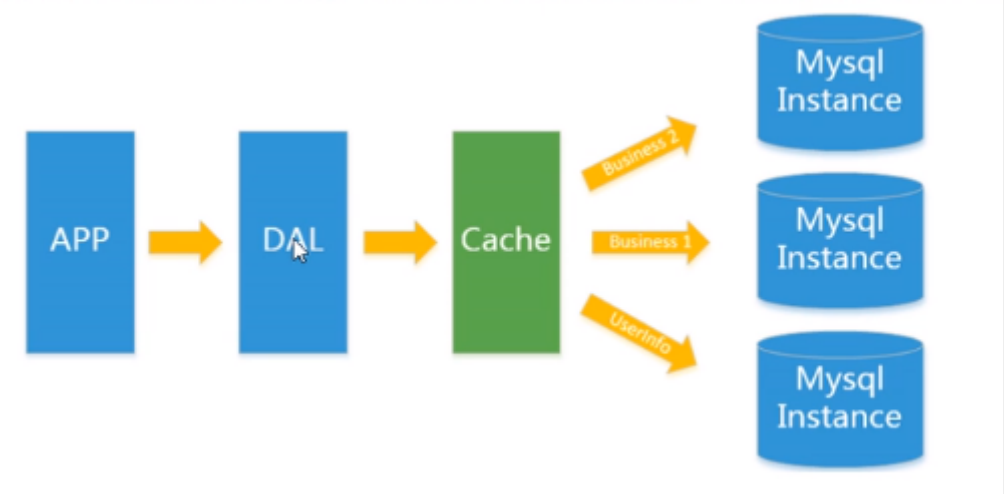


上述架构下，我们来看看数据存储的瓶颈是什么?

1. 数据量的总大小 一个机器放不下时
2. 数据的索引（B+ Tree）一个机器的内存放不下时
3. 访问量(读写混合)**一个实例**不能承受

1.2 Memcached(缓存)+MySQL+垂直拆分

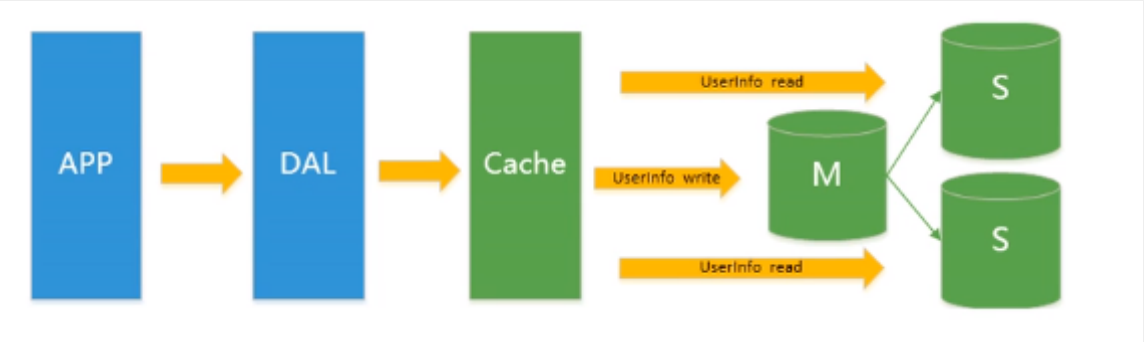
后来，随着访问量的上升，几乎大部分使用MySQL架构的网站在数据库上都开始出现了性能问题，web程序不再仅仅专注在功能上，同时也在追求性能。程序员们开始大量的使用缓存技术来缓解数据库的压力，优化数据库的结构和索引。开始比较流行的是通过文件缓存来缓解数据库压力，但是当访问量继续增大的时候，多台web机器通过文件缓存不能共享，大量的小文件缓存也带了了**比较高的IO压力**。在这个时候，Memcached就自然的成为一个非常时尚的技术产品。



Memcached作为一个独立的分布式的缓存服务器，为多个web服务器提供了一个共享的高性能缓存服务，在Memcached服务器上，又发展了根据hash算法来进行多台Memcached缓存服务的扩展，然后又出现了一致性hash来解决增加或减少缓存服务器导致重新hash带来的大量缓存失效的弊端。

1.3 MySql主从读写分离

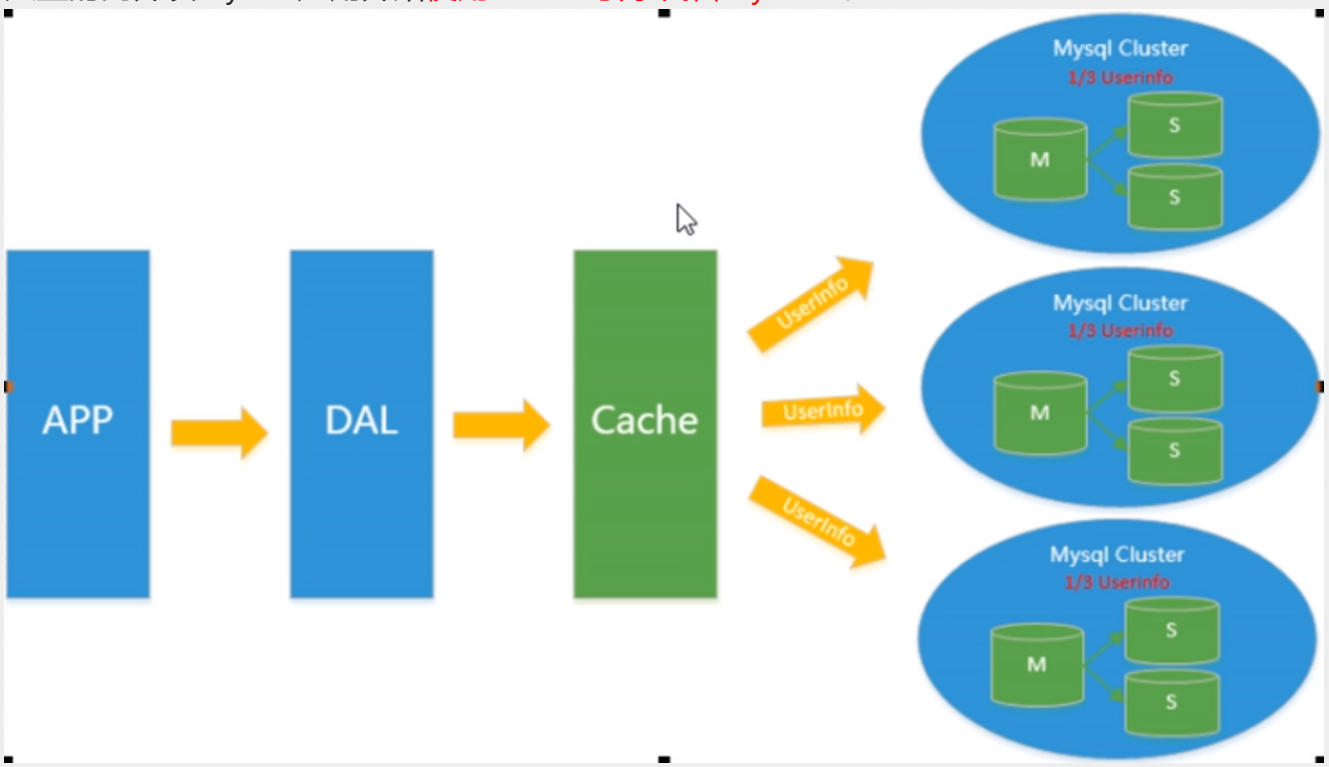
由于数据库的写入压力增加，Memcached只能缓解数据库的读取压力。读写集中在一个数据库上让数据库不堪重负，大部分网站开始使用主从复制技术来达到读写分离，以提高读写性能和读库的可扩展性。Mysql的master-slave模式成为这个时候的网站标配了。



1.4 分表分库+水平拆分+MySql集群

在Memcached的高速缓存，MySQL的主从复制，读写分离的基础之上，这时MySQL主库的写压力开始出现瓶颈，而数据量的持续猛增，由于MyISAM使用表锁，在高并发下会出现严重的**锁问题**，

大量的高并发MySQL应用开始使用InnoDB引擎代替MyISAM。

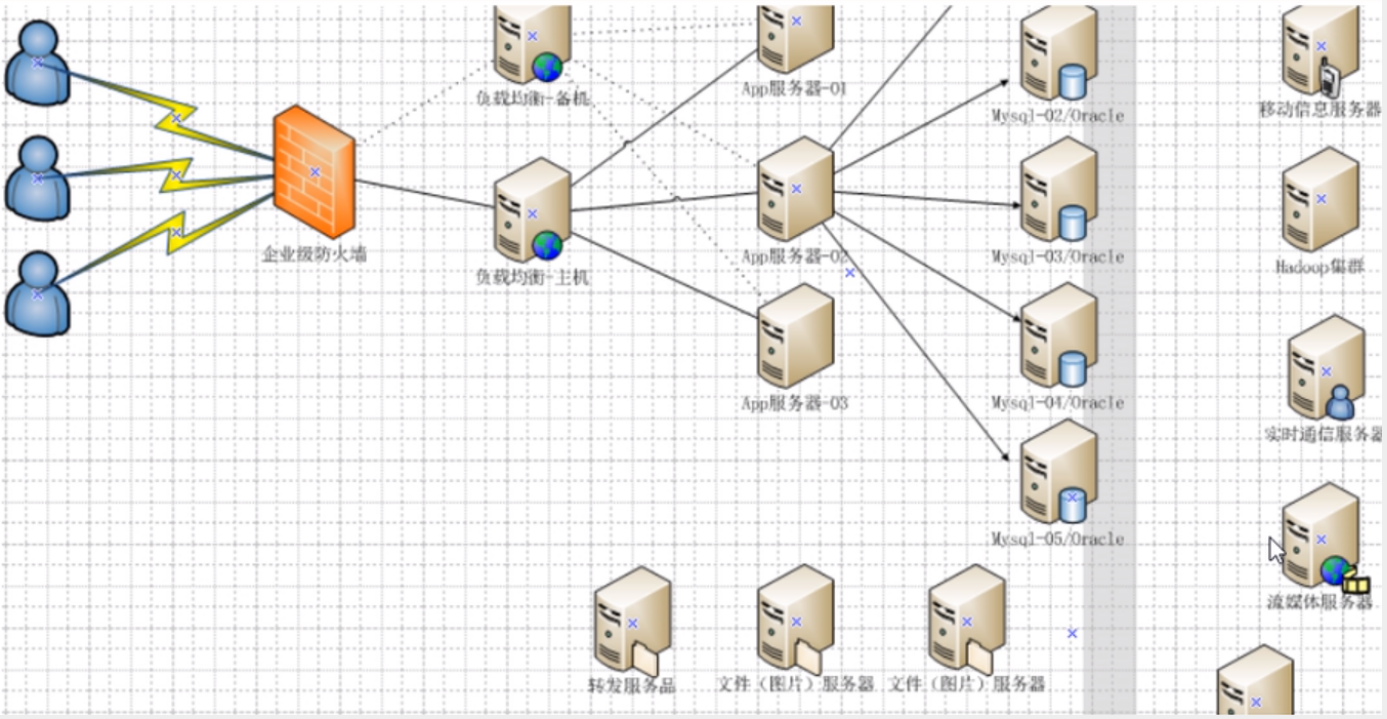


同时，开始流行使用分表分库来缓解写压力和数据增长的扩展问题。这个时候，分表分库成了一个热门技术，是面试的热门问题也是业界讨论的热门技术问题。也就在这个时候，MySQL推出了**还不太稳定的表分区**，这也给技术实力一般的公司带来了希望。**虽然MySQL推出了MySQL Cluster集群，但性能也不能很好满足互联网的要求**，只是在高可靠性上提供了非常大的保证。

1.5 MySql的扩展性瓶颈

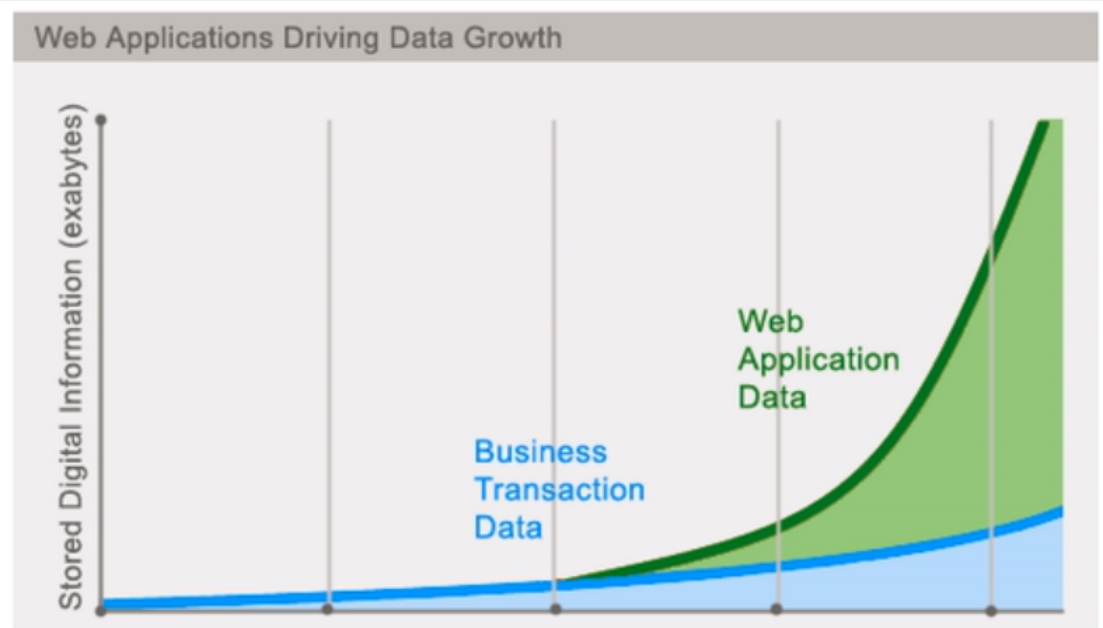
MySQL数据库也经常存储一些大文本字段，导致数据库表非常的大，在做数据库恢复的时候就导致非常的慢，不容易快速恢复数据库。比如1000万4KB大小的文本就接近40GB的大小，**如果能把这些数据从MySQL省去，MySQL将变得非常的小**。关系数据库很强大，但是它并不能很好的应付所有的应用场景。MySQL的扩展性差（需要复杂的技术来实现），大数据下IO压力大，表结构更改困难，正是当前使用MySQL的开发人员面临的问题。

1.6 今天是怎么样子



1.7 为什么用NoSql

今天我们可以通过第三方平台（如：Google,Facebook等）可以很容易的访问和抓取数据。用户的个人信息，社交网络，地理位置，用户生成的数据和用户操作日志已经成倍的增加。我们如果要对这些用户数据进行挖掘，那SQL数据库已经不适合这些应用了, NoSQL数据库的发展也却能很好的处理这些大的数据。



2. 什么是NoSql

NoSQL(NoSQL = Not Only SQL)，意即“不仅仅是SQL”

泛指非关系型的数据库。随着互联网web2.0网站的兴起，传统的关系数据库在应付web2.0网站，特别是超大规模和高并发的SNS类型的web2.0纯动态网站已经显得力不从心，暴露了很多难以克服的问题，而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。NoSQL数据库的产生

就是为了解决大规模数据集合多重数据种类带来的挑战，尤其是大数据应用难题，包括超大规模数据的存储。（例如谷歌或Facebook每天为他们的用户收集万亿比特的数据）。这些类型的数据存储不需要固定的模式，无需多余操作就可以横向扩展。

3. 能干啥

3.1 易扩展

NoSQL数据库种类繁多，但是一个共同的特点都是去掉关系数据库的关系型特性。
数据之间无关系，这样就非常容易扩展。也无形之间，在架构的层面上带来了可扩展的能力。

3.2 大数据高性能

NoSQL数据库都具有非常高的读写性能，尤其在大数据量下，同样表现优秀。
这得益于它的无关系性，数据库的结构简单。
一般MySQL使用Query Cache，每次表的更新Cache就失效，是一种大粒度的Cache 在针对web2.0的交互频繁的应用，Cache性能不高。而NoSQL的Cache是记录级的，是一种细粒度的Cache，所以NoSQL在这个层面上来说就要性能高很多了

3.3 灵活的数据模型

NoSQL无需事先为要存储的数据建立字段，随时可以存储自定义的数据格式。
而在关系数据库里，增删字段是一件非常麻烦的事情。如果是非常大数据量的表，增加字段简直就是一个噩梦。

3.4 传统RDBMS VS NOSQL

RDBMS

- **高度组织化结构化数据**
- 结构化查询语言（SQL）
- 数据和关系都存储在单独的表中。
- 数据操纵语言，数据定义语言
- 严格的一致性
- 基础事务

NoSQL

- 代表着不仅仅是SQL
- 没有声明性查询语言
- 没有预定义的模式 -键 - 值对存储，列存储，文档存储，图形数据库
- 最终一致性，而非ACID属性
- 非结构化和不可预知的数据

- CAP定理
- 高性能，高可用性和可伸缩性

4. 3V与3高

4.1 大数据时代的3V

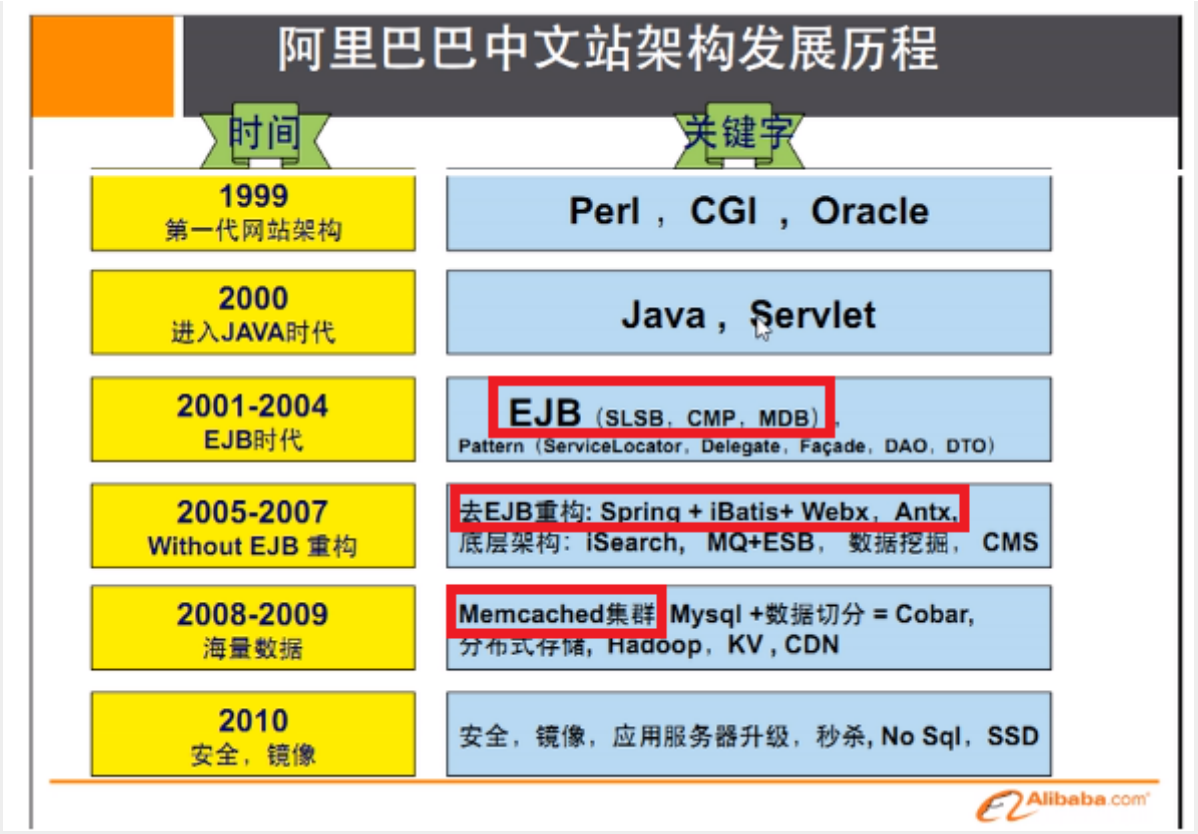
- 海量 Volume
- 多样 Variety
- 实时 Velocity

4.2 互联网需求的3高

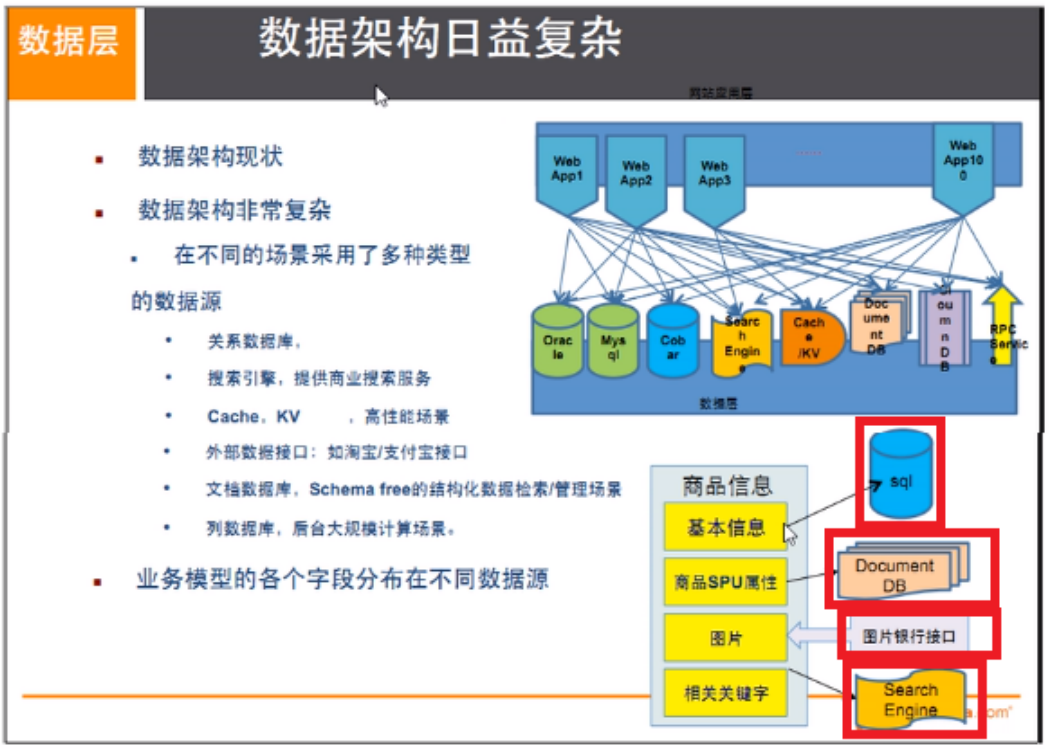
- 高并发
- 高可扩
- 高性能

5. 应用：以阿里巴巴中文网站为例

5.1 架构发展历程



5.2 多数据源和多数据类型存储相关



5.2.1 商品基本信息

- 名称、价格、出厂日期、生产厂商等
  关系型数据库：mysql/oracle目前淘宝在**去O化**(也即拿掉Oracle)，
  注意，淘宝内部用的Mysql是里面的大牛自己改造过的
- 去IOE
  2008年，王坚加盟阿里巴巴成为集团首席架构师，即现在的首席技术官。这位前微软亚洲研究院常务副院长被马云定位为：将帮助阿里巴巴集团建立世界级的技术团队，并负责集团技术架构以及基础技术平台搭建。
  在加入阿里后，带着技术基因和学者风范的王坚就在阿里巴巴集团提出了被称为"去IOE"（在IT建设过程中，去除IBM小型机、Oracle数据库及EMC存储设备）的想法，并开始把云计算的本质，植入阿里IT基因。 王坚这样概括"去IOE"运动和阿里云之间的关系："去IOE"彻底改变了阿里集团IT架构的基础，是阿里拥抱云计算，产出计算服务的基础。"去IOE"的本质是分布化，让随处可以买到的Commodity PC架构成为可能，使云计算能够落地的首要条件。

5.2.2 商品详细信息

- 多文字信息描述类，IO读写性能变差
- 文档数据库MongDB中

5.2.3 商品图片

分布式的文件系统

- 淘宝自己的TFS
- Google的GFS

- Hadoop的HDFS

5.2.4 商品的关键字

搜索引擎，淘宝内用，ISearch

5.2.5 商品波段性的热点高频信息

内存数据库
tair、redis、Memcache

5.2.6 商品的交易、价格计算、积分累计

外部系统，外部第三方支付接口
支付宝

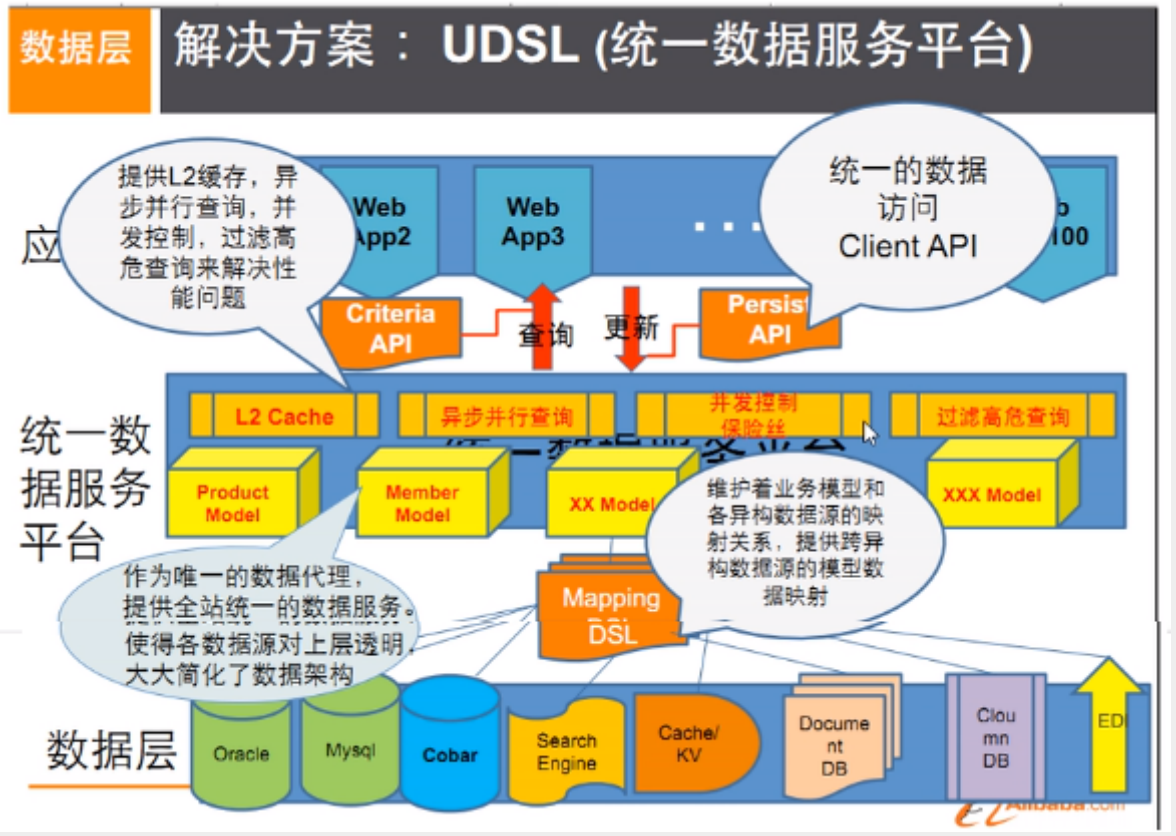
5.3 总结大型互联网应用(大数据、高并发、多样数据类型)的难点和解决方案

5.3.1 难点

- 数据多样性
- 数据源多样性
- 数据源改造而服务平台不需要大规模重构

5.3.2 解决办法

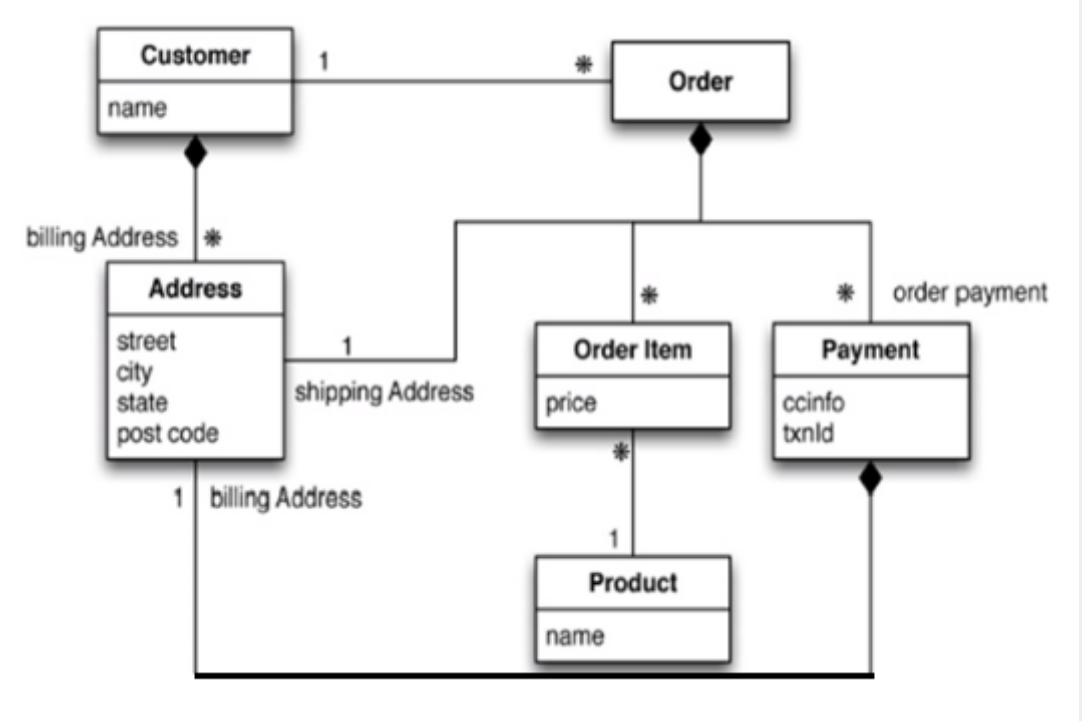
统一数据平台服务层UDSL



6. NoSql数据模型简介

以一个电商客户、订单、订购、地址模型来对比下关系型数据库和非关系型数据库为例

6.1 传统关系型数据库模型



6.2 NoSql设计

BSON - 是一种类json的一种二进制形式的存储格式，简称 Binary JSON，和JSON一样支持内嵌的文档对象和数组。

```
{
 "customer":{
  "id":1136,
  "name":"Z3",
  "billingAddress":[{"city":"beijing"}],
  "orders":[
   {
    "id":17,
    "customerId":1136,
    "orderItems":[{"productId":27,"price":77.5,"productName":"thinking in java"}],
    "shippingAddress":[{"city":"beijing"}]

"orderPayment":[{"ccinfo":"111-222-333","txnid":"asdfadcd334","billingAddress":{"city":"beiji
ng"}}],
    }
   ]
  }
}
```

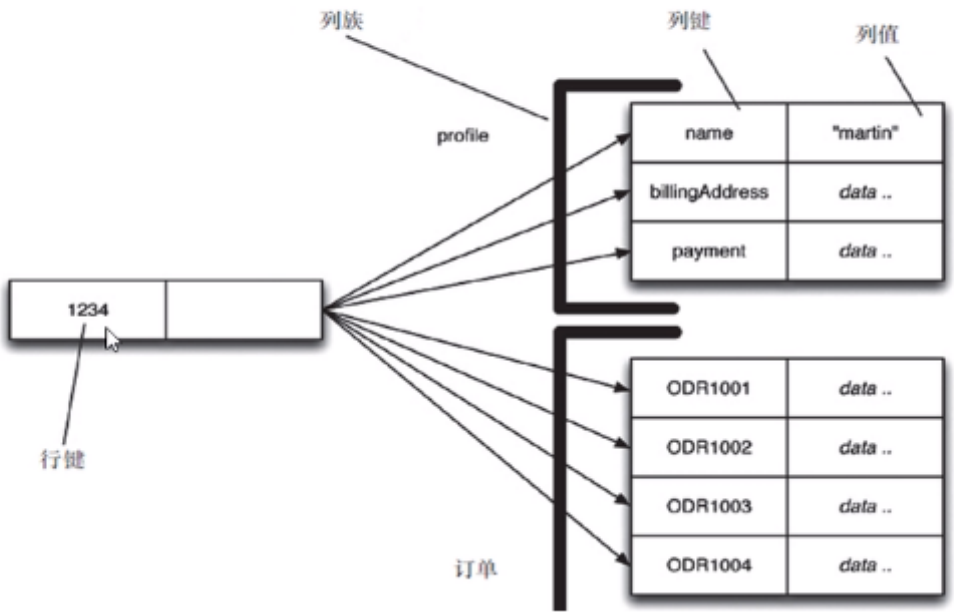
Nosql - 聚合模型
分布式事务是支持不了太多的并发的

6.3 NoSql的聚合模型

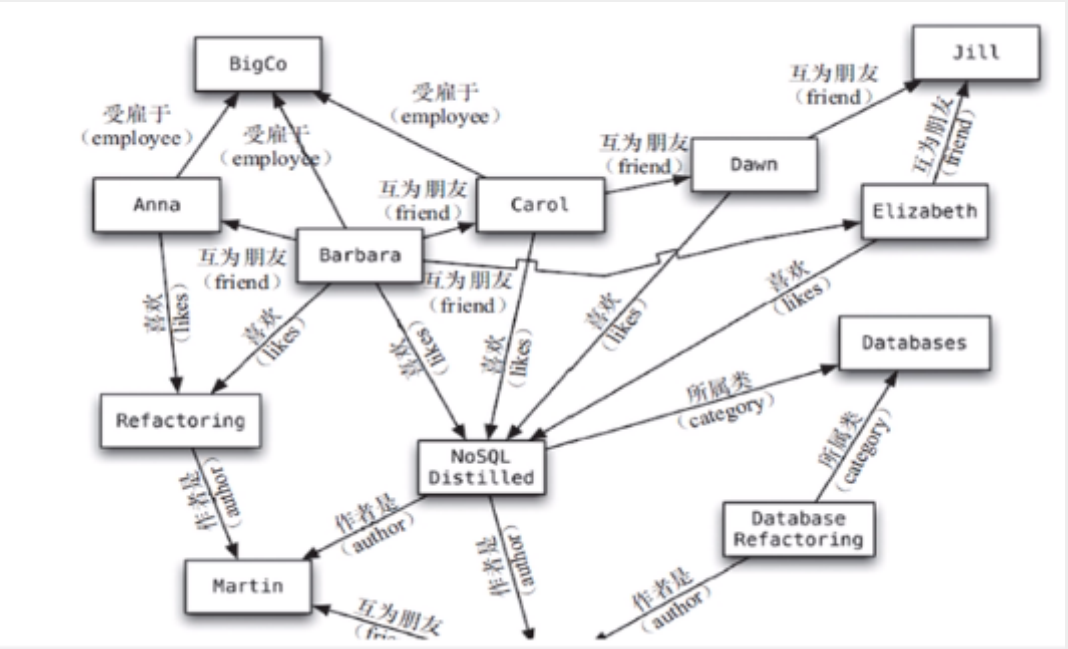
- KV键值对
- Bson

- 列族

顾名思义，是按列存储数据的。最大的特点是方便存储结构化和半结构化数据，方便做数据压缩，对针对某一列或者某几列的查询有非常大的IO优势。



- 图形



7. NoSql数据库的分类

7.1 KV键值对

- 新浪微博：BerkeleyDB+Redis
- 美团：Redis+tair
- 阿里、百度：memcache+Redis

7.2 文档型数据库(bson格式比较多)

- CouchDB
- MongoDB
  MongoDB 是一个基于分布式文件存储的数据库。由 C++ 语言编写。旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。 MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。

7.3 列存储数据库

- Cassandra, HBase
- 分布式文件系统

7.4 图关系数据库

- 它不是放图形的，放的是关系比如:朋友圈社交网络、广告推荐系统
- 社交网络，推荐系统等。专注于构建关系图谱
- Neo4J, InfoGrid

7.5 四者对比

分类	Examples举例	典型应用场景	数据模型	优点	缺点
键值（key-value）[3]	Tokyo Cabinet/Tyrant, Redis, Voldemort, Oracle BDB	内容缓存，主要用于处理大量数据的高访问负载，也用于一些日志系统等等。[3]	Key 指向 Value 的键值对，通常用hash table来实现 [3]	查找速度快	数据无结构化，通常只被当作字符串或者二进制数据 [3]
列存储数据库 [3]	Cassandra, HBase, Riak	分布式的文件系统	以列簇式存储，将同一列数据存在一起	查找速度快，可扩展性强，更容易进行分布式扩展	功能相对局限
文档型数据库 [3]	CouchDB, MongoDb	Web应用（与Key-Value类似，Value是结构化的，不同的是数据库能够了解Value的内容）	Key-Value对应的键值对，Value为结构化数据	数据结构要求不严格，表结构可变，不需要像关系型数据库一样需要预先定义表结构	查询性能不高，而且缺乏统一的查询语法。
图形(Graph)数据库 [3]	Neo4J, InfoGrid, Infinite Graph	社交网络，推荐系统等。专注于构建关系图谱	图结构	利用图结构相关算法。比如最短路径寻址，N度关系查	很多时候需要对整个图做计算才能得出需要的信息，而且这种结

8. CAP原理与BASE

8.1 传统的ACID

- A (Atomicity) 原子性
- C (Consistency) 一致性
- I (Isolation) 独立性
- D (Durability) 持久性

8.2 NoSql的CAP

- **C**:Consistency（强一致性）
- **A**:Availability（可用性）
- **P**:Partition tolerance（分区容错性）

8.3 CAP的3进2

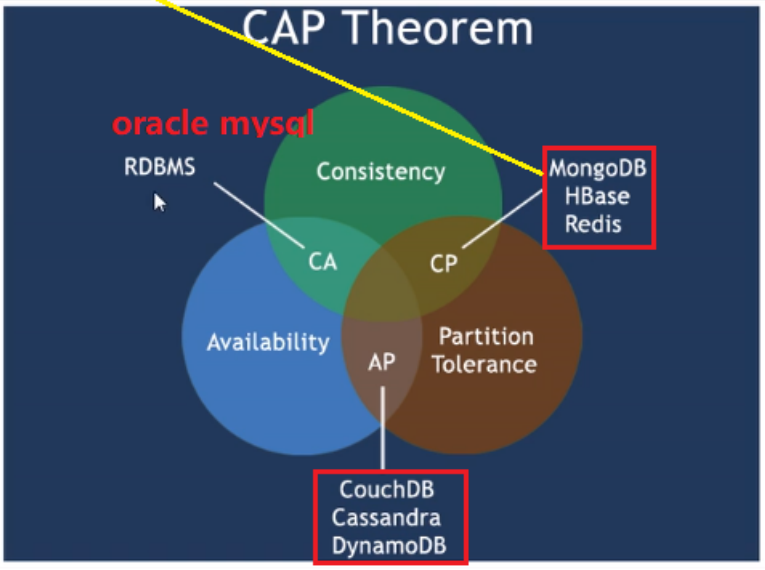
CAP理论的核心是：一个分布式系统不可能同时很好的满足一致性，可用性和分区容错性这三个需求，最多只能同时较好的满足两个。

因此，根据 CAP 原理将 NoSQL 数据库分成了满足 CA 原则、满足 CP 原则和满足 AP 原则三 大类：

CA - 单点集群，满足一致性，可用性的系统，通常在可扩展性上不太强大。

CP - 满足一致性，分区容忍必的系统，通常性能不是特别高。

AP - 满足可用性，分区容忍性的系统，通常可能对一致性要求低一些。



分区容忍性是我们必须需要实现的。

所以我们只能在一致性和可用性之间进行权衡，*没有NoSQL系统能同时保证这三点。*

C:强一致性 A：高可用性 P：分布式容忍性

CA 传统Oracle数据库

AP 大多数网站架构的选择

CP Redis、Mongodb

注意：分布式架构的时候必须做出取舍。

8.4 BASE

BASE就是为了解决关系数据库强一致性引起的问题而引起的可用性降低而提出的解决方案。
BASE其实是下面三个术语的缩写：

- 基本可用（Basically Available）
- 软状态（Soft state）
- 最终一致（Eventually consistent）
  它的思想是通过让系统放松对某一时刻**数据一致性**的要求来换取系统整体伸缩性和性能上改观。为什么这么说呢，缘由就在于大型系统往往由于**地域分布和极高性能**的要求，不可能采用分布式事务来完成这些指标，要想获得这些指标，我们必须采用另外一种方式来完成，这里BASE就是解决这个问题的办法