Atitit.nosql api 标准化 以及nosql数据库的实现模型分类差异

[1. 常用的nosql数据库MongoDB Cassandra 1](#_Toc572)

[1.1. 查询> db.blogposts.find( { "author.name" : "Jane" } ) 1](#_Toc9877)

[1.2. 保存save insert 1](#_Toc20513)

[1.3. Update ，delete 2](#_Toc1832)

[1.4. MongoDB 与 RDBMS Where 语句比较 2](#_Toc8385)

[1.5. MongoDB OR 条件 2](#_Toc19158)

[1.5.1. 实例 3](#_Toc3881)

[1.6. AND 和 OR 联合使用 3](#_Toc17648)

[2. 对比 4](#_Toc31752)

[3. 参考 5](#_Toc27491)

# Keyword

Zai json中实现表达式的表达，用json来表示sql的表达式

# 常用的nosql数据库****MongoDB Cassandra****

举例2：在一个关系型数据库中，一篇博客（包含文章内容、评论、评论的投票）会被打散在多张数据表中。在MongoDB中，能用一个文档来表示一篇博客， 评论与投票作为文档数组，放在正文主文档中。这样数据更易于管理，消除了传统关系型数据库中影响性能和水平扩展性的“JOIN”操作。

## 查询> db.blogposts.find( { "author.name" : "Jane" } )

## 保存save insert

> db.blogposts.save({ title : "My First Post", author: {name : "Jane", id :1},  
  comments : [{ by: "Abe", text: "First" },  
              { by : "Ada", text : "Good post" }]  
})

## Update ，delete

作者:: 绰号:老哇的爪子 （ 全名：：Attilax Akbar Al Rapanui 阿提拉克斯 阿克巴 阿尔 拉帕努伊 ） 汉字名：艾龙，  EMAIL:1466519819@qq.com

转载请注明来源： http://blog.csdn.net/attilax

## MongoDB 与 RDBMS Where 语句比较

如果你熟悉常规的 SQL 数据，通过下表可以更好的理解 MongoDB 的条件语句查询：

| **操作** | **格式** | **范例** | **RDBMS中的类似语句** |
| --- | --- | --- | --- |
| 等于 | {<key>:<value>} | db.col.find({"by":"菜鸟教程"}).pretty() | where by = '菜鸟教程' |
| 小于 | {<key>:{$lt:<value>}} | db.col.find({"likes":{$lt:50}}).pretty() | where likes < 50 |
| 小于或等于 | {<key>:{$lte:<value>}} | db.col.find({"likes":{$lte:50}}).pretty() | where likes <= 50 |
| 大于 | {<key>:{$gt:<value>}} | db.col.find({"likes":{$gt:50}}).pretty() | where likes > 50 |
| 大于或等于 | {<key>:{$gte:<value>}} | db.col.find({"likes":{$gte:50}}).pretty() | where likes >= 50 |
| 不等于 | {<key>:{$ne:<value>}} | db.col.find({"likes":{$ne:50}}).pretty() | where likes != 50 |

## MongoDB OR 条件

MongoDB OR 条件语句使用了关键字 **$or**,语法格式如下：

>db.col.find(

{

$or: [

{key1: value1}, {key2:value2}

]

}).pretty()

### **实例**

以下实例中，我们演示了查询键 **by** 值为 菜鸟教程 或键 **title** 值为 **MongoDB 教程** 的文档。

>db.col.find({$or:[{"by":"菜鸟教程"},{"title": "MongoDB 教程"}]}).pretty(){

"\_id" : ObjectId("56063f17ade2f21f36b03133"),

"title" : "MongoDB 教程",

"description" : "MongoDB 是一个 Nosql 数据库",

"by" : "菜鸟教程",

"url" : "http://www.runoob.com",

"tags" : [

"mongodb",

"database",

"NoSQL"

],

"likes" : 100}>

## AND 和 OR 联合使用

以下实例演示了 AND 和 OR 联合使用，类似常规 SQL 语句为： **'where url='http://www.runoob.com' AND (by = '菜鸟教程' OR title = 'MongoDB 教程')'**

>db.col.find({"likes": {$gt:50}, $or: [{"by": "菜鸟教程"},{"title": "MongoDB 教程"}]}).pretty(){

"\_id" : ObjectId("56063f17ade2f21f36b03133"),

"title" : "MongoDB 教程",

"description" : "MongoDB 是一个 Nosql 数据库",

"by" : "菜鸟教程",

"url" : "http://www.runoob.com",

"tags" : [

"mongodb",

"database",

"NoSQL"

],

"likes" : 100}

# 对比

Cassandra 教程资料较为少，crud语句类似sql了。。不是nosql了。。

MongoDB的资料多，更加容易oo

不过百度查询MongoDB 百度为您找到相关结果约3,110,000个

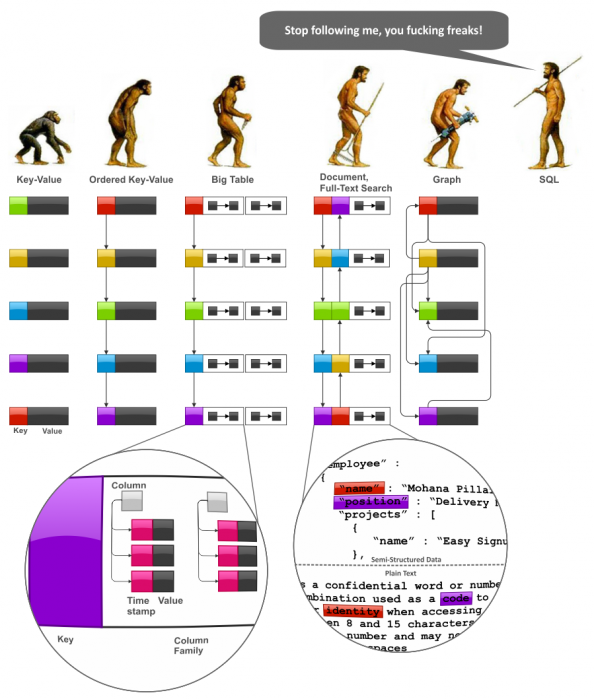
Cassandra 的数量是mongodb的俩被。。

Hbase 百度为您找到相关结果约6,110,000个

至于hbase ，文档少，貌似比较复杂不好用。

# 下图是NoSQL家族的进化图，我们可以看到这样的进化：Key-Value时代，BigTable时代，Document时代，全文搜索时代，和Graph数据库时代：

（陈皓注：注意图中SQL说的那句话，NoSQL再这样发展下去就是SQL了，哈哈。）

[](http://cms.csdnimg.cn/articlev1/uploads/allimg/120515/09393Q0D-0.png)

数据模型的进化：

* **Key-Value键值对存储**是非常简单而强大的。下面的很多技术基本上都是基于这个技术开始发展的。但是，Key-Value有一个非常致命的问题，那就是如果我们需要查找一段范围内的key。（陈皓注：学过hash-table数据结构的人都应该知道，hash-table是非序列容器，其并不像数组，链接，队列这些有序容器，我们可以控制数据存储的顺序）。于是，有序键值（Ordered Key-Value）数据模型被设计出来解决这一限制，来从根本上提高数据集的问题。
* **Ordered Key-Value有序键值**模型也非常强大，但是，其也没有对Value提供某种数据模型。通常来说，Value的模型可以由应用负责解析和存取。这种很不方便，于是出现了BigTable类型的数据库，这个数据模型其实就是map里有map，map里再套map，一层一层套下去，也就是层层嵌套的key- value（value里又是一个key-value），这种数据库的Value主要通过“列族”（column families），列，和时间截来控制版本。（陈皓注：关于时间截来对数据的版本控制主要是解决数据存储并发问题，也就是所谓的乐观锁，详见《[多版本并发控制(MVCC)在分布式系统中的应用](http://coolshell.cn/articles/6790.html" \o "多版本并发控制(MVCC)在分布式系统中的应用" \t "http://www.csdn.net/article/2012-05-15/_blank)》）
* **Document databases 文档数据库** 改进了BigTable模型，并提供了两个有意义的改善。第一个是允许Value中有主观的模式（scheme），而不是map套map。第二个是索引。**Full Text Search Engines全文搜索引擎**可以被看作是文档数据库的一个变种，他们可以提供灵活的可变的数据模式（scheme）以及自动索引。他们之间的不同点主要是，文档数据库用字段名做索引，而全文搜索引擎用字段值做索引。
* **Graph data models图式数据库** 可以被认为是这个进化过程中从Ordered Key-Value数据库发展过来的一个分支。图式数据库允许构建议图结构的数据模型。它和文档数据库有关系的原因是，它的很多实现允许value可以是一个map或是一个document。

#### **NoSQL数据模型摘要**

本文剩下的章节将向你介绍数据建模的技术实现和相关模式。但是，在介绍这些技术之前，先来一段序言：

* NoSQL数据模型设计一般从业务应用的具体数据查询入手，而不是数据间的关系：
* 关系型的数据模型基本上是分析数据间的结构和关系。其设计理念是： “**What answers do I have?”**
* NoSQL数据模型基本上是从应用对数据的存取方式入手，如：我需要支持某种数据查询。其设计理念是**”What questions do I have?”**
* NoSQL数据模型设计比关系型数据库需要对数据结构和算法的更深的了解。在这篇文章中我会和大家说那些尽人皆知的数据结构，这些数据结构并不只是被NoSQL使用，但是对于NoSQL的数据模型却非常有帮助。
* 数据冗余和反规格化是一等公民。
* 关系型数据库对于处理层级数据和图式数据非常的不方便。NoSQL用来解决图式数据明显是一个非常好的解决方案，几乎所有的NoSQL数据库可以很强地解决此类问题。这就是为什么这篇文章专门拿出一章来说明层级数据模型。

## 下面是NoSQL的分类表，也是我用来写这篇文章时做实践的产品：

* Key-Value 存储: Oracle Coherence, Redis, Kyoto Cabinet
* 类BigTable存储: Apache HBase, Apache Cassandra
* 文档数据库: MongoDB, CouchDB
* 全文索引: Apache Lucene, Apache Solr
* 图数据库: neo4j, FlockDB

# 参考

排名前十的SQL和NoSQL数据库-CSDN.NET.htm

8天学通MongoDB——第一天 基础入门 - 一线码农 - 博客园.htm

MongoDB 查询文档 菜鸟教程.htm

分布式 Key-Value 存储系统：Cassandra 入门.htm

Cassandra 2.x中文教程（2）：执行简单的CQL操作 屁民部落.htm

NoSQL数据建模技术-CSDN.NET.htm

HBase使用教程 - 推酷.htm