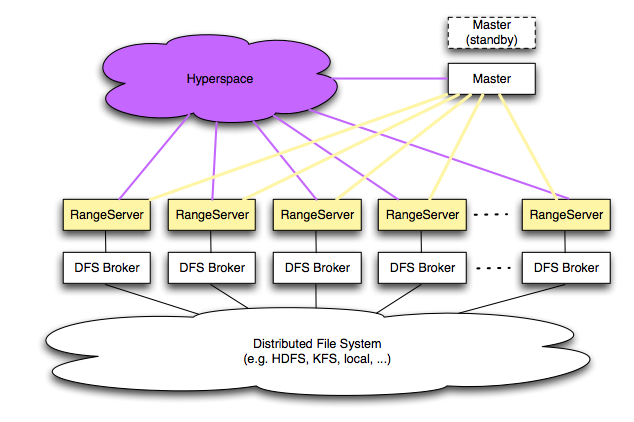
**分布式数据库 Hypertable**

Hypertable是一个开源、高性能、可伸缩的数据库，它采用与Google的Bigtable相似的模型。在过去数年中，Google为在PC集群 上运行的可伸缩计算基础设施设计建造了三个关键部分。第一个关键的基础设施是Google File System（GFS），这是一个高可用的文件系统，提供了一个全局的命名空间。它通过跨机器（和跨机架）的文件数据复制来达到高可用性，并因此免受传统 文件存储系统无法避免的许多失败的影响，比如电源、内存和网络端口等失败。第二个基础设施是名为Map-Reduce的计算框架，它与GFS紧密协作，帮 助处理收集到的海量数据。第三个基础设施是Bigtable，它是传统数据库的替代。Bigtable让你可以通过一些主键来组织海量数据，并实现高效的 查询。Hypertable是Bigtable的一个开源实现，并且根据我们的想法进行了一些改进。

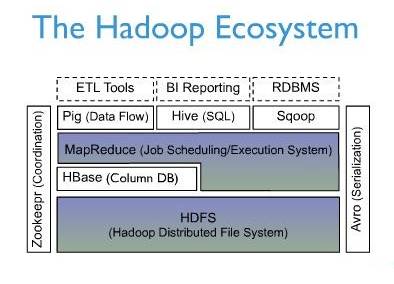


* **授权协议：**[GPLv2](http://www.oschina.net/question/12_2826)
* **开发语言：**[C/C++](http://www.oschina.net/project/lang/21/c)
* **操作系统： Linux  Mac OS**

[**分布式数据库 HBase**](http://www.oschina.net/p/hbase)

HBase – Hadoop Database，是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统，利用HBase技术可在廉价PC Server上搭建起大规模结构化存储集群。

HBase是Google Bigtable的开源实现，类似Google Bigtable利用GFS作为其文件存储系统，HBase利用Hadoop HDFS作为其文件存储系统；Google运行MapReduce来处理Bigtable中的海量数据，HBase同样利用Hadoop MapReduce来处理HBase中的海量数据；Google Bigtable利用 Chubby作为协同服务，HBase利用Zookeeper作为对应。

[](http://www.searchtb.com/wp-content/uploads/2011/01/image0010.jpg)

上图描述了Hadoop EcoSystem中的各层系统，其中HBase位于结构化存储层，Hadoop HDFS为HBase提供了高可靠性的底层存储支持，Hadoop MapReduce为HBase提供了高性能的计算能力，Zookeeper为HBase提供了稳定服务和failover机制。

此外，Pig和Hive还为HBase提供了高层语言支持，使得在HBase上进行数据统计处理变的非常简单。 Sqoop则为HBase提供了方便的RDBMS数据导入功能，使得传统数据库数据向HBase中迁移变的非常方便。

# HBase访问接口

1.       Native Java API，最常规和高效的访问方式，适合Hadoop MapReduce Job并行批处理HBase表数据

2.       HBase Shell，HBase的命令行工具，最简单的接口，适合HBase管理使用

3.       Thrift Gateway，利用Thrift序列化技术，支持C++，PHP，Python等多种语言，适合其他异构系统在线访问HBase表数据

4.       REST Gateway，支持REST 风格的Http API访问HBase, 解除了语言限制

5.       Pig，可以使用Pig Latin流式编程语言来操作HBase中的数据，和Hive类似，本质最终也是编译成MapReduce Job来处理HBase表数据，适合做数据统计

6.       Hive，当前Hive的Release版本尚没有加入对HBase的支持，但在下一个版本Hive 0.7.0中将会支持HBase，可以使用类似SQL语言来访问HBase

# HBase数据模型

## Table & Column Family

|  |  |  |  |
| --- | --- | --- | --- |
| **Row Key** | **Timestamp** | **Column Family** | |
| **URI** | **Parser** |
| r1 | t3 | url=http://www.taobao.com | title=天天特价 |
| t2 | host=taobao.com |  |
| t1 |  |  |
| r2 | t5 | url=http://www.alibaba.com | content=每天… |
| t4 | host=alibaba.com |  |

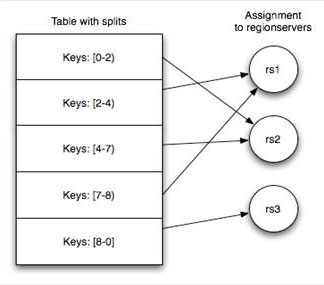
Ø  Row Key: 行键，Table的主键，Table中的记录按照Row Key排序

Ø  Timestamp: 时间戳，每次数据操作对应的时间戳，可以看作是数据的version number

Ø  Column Family：列簇，Table在水平方向有一个或者多个Column Family组成，一个Column Family中可以由任意多个Column组成，即Column Family支持动态扩展，无需预先定义Column的数量以及类型，所有Column均以二进制格式存储，用户需要自行进行类型转换。

## Table & Region

当Table随着记录数不断增加而变大后，会逐渐分裂成多份splits，成为regions，一个region由[startkey,endkey)表示，不同的region会被Master分配给相应的RegionServer进行管理：

[](http://www.searchtb.com/wp-content/uploads/2011/01/image0021.jpg)

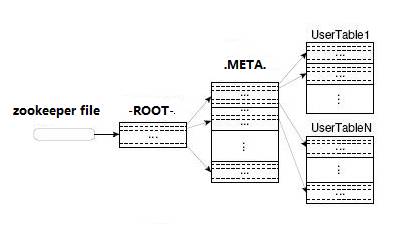
## -ROOT- && .META. Table

HBase中有两张特殊的Table，-ROOT-和.META.

Ø  .META.：记录了用户表的Region信息，.META.可以有多个regoin

Ø  -ROOT-：记录了.META.表的Region信息，-ROOT-只有一个region

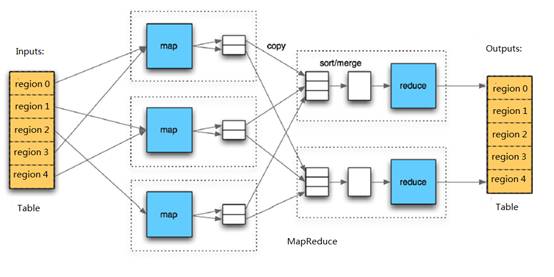
Ø  Zookeeper中记录了-ROOT-表的location

[](http://www.searchtb.com/wp-content/uploads/2011/01/image0030.jpg)

Client访问用户数据之前需要首先访问zookeeper，然后访问-ROOT-表，接着访问.META.表，最后才能找到用户数据的位置去访问，中间需要多次网络操作，不过client端会做cache缓存。

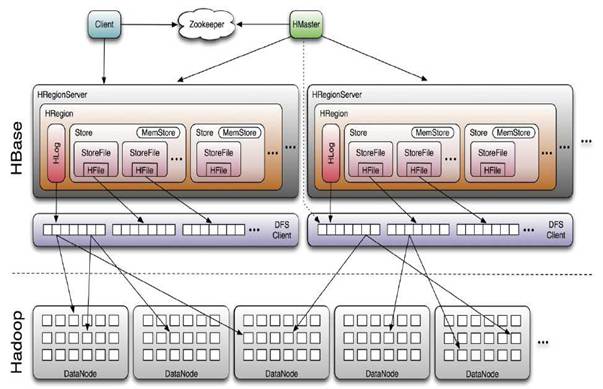
# MapReduce on HBase

在HBase系统上运行批处理运算，最方便和实用的模型依然是MapReduce，如下图：

[](http://www.searchtb.com/wp-content/uploads/2011/01/image0040.jpg)

HBase Table和Region的关系，比较类似HDFS File和Block的关系，HBase提供了配套的TableInputFormat和TableOutputFormat API，可以方便的将HBase Table作为Hadoop MapReduce的Source和Sink，对于MapReduce Job应用开发人员来说，基本不需要关注HBase系统自身的细节。

# HBase系统架构

[](http://www.searchtb.com/wp-content/uploads/2011/01/image0050.jpg)

## Client

HBase Client使用HBase的RPC机制与HMaster和HRegionServer进行通信，对于管理类操作，Client与HMaster进行RPC；对于数据读写类操作，Client与HRegionServer进行RPC

## Zookeeper

Zookeeper Quorum中除了存储了-ROOT-表的地址和HMaster的地址，HRegionServer也会把自己以Ephemeral方式注册到 Zookeeper中，使得HMaster可以随时感知到各个HRegionServer的健康状态。此外，Zookeeper也避免了HMaster的 单点问题，见下文描述

## HMaster

HMaster没有单点问题，HBase中可以启动多个HMaster，通过Zookeeper的Master Election机制保证总有一个Master运行，HMaster在功能上主要负责Table和Region的管理工作：

1.       管理用户对Table的增、删、改、查操作

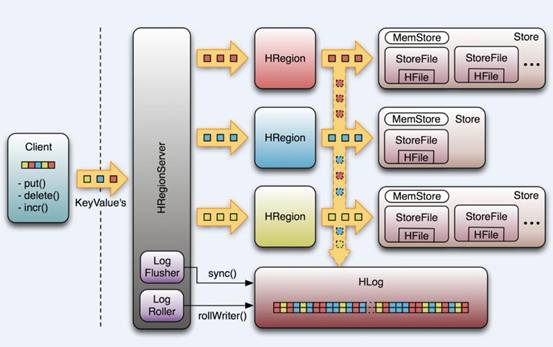
2.       管理HRegionServer的负载均衡，调整Region分布

3.       在Region Split后，负责新Region的分配

4.       在HRegionServer停机后，负责失效HRegionServer 上的Regions迁移

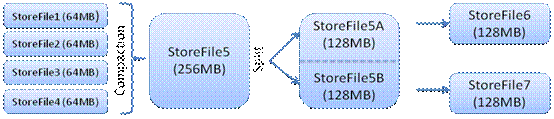
## HRegionServer

HRegionServer主要负责响应用户I/O请求，向HDFS文件系统中读写数据，是HBase中最核心的模块。

[](http://www.searchtb.com/wp-content/uploads/2011/01/image0060.jpg)

HRegionServer内部管理了一系列HRegion对象，每个HRegion对应了Table中的一个Region，HRegion中由多 个HStore组成。每个HStore对应了Table中的一个Column Family的存储，可以看出每个Column Family其实就是一个集中的存储单元，因此最好将具备共同IO特性的column放在一个Column Family中，这样最高效。

HStore存储是HBase存储的核心了，其中由两部分组成，一部分是MemStore，一部分是StoreFiles。MemStore是 Sorted Memory Buffer，用户写入的数据首先会放入MemStore，当MemStore满了以后会Flush成一个StoreFile（底层实现是HFile）， 当StoreFile文件数量增长到一定阈值，会触发Compact合并操作，将多个StoreFiles合并成一个StoreFile，合并过程中会进 行版本合并和数据删除，因此可以看出HBase其实只有增加数据，所有的更新和删除操作都是在后续的compact过程中进行的，这使得用户的写操作只要 进入内存中就可以立即返回，保证了HBase I/O的高性能。当StoreFiles Compact后，会逐步形成越来越大的StoreFile，当单个StoreFile大小超过一定阈值后，会触发Split操作，同时把当前 Region Split成2个Region，父Region会下线，新Split出的2个孩子Region会被HMaster分配到相应的HRegionServer 上，使得原先1个Region的压力得以分流到2个Region上。下图描述了Compaction和Split的过程：

[](http://www.searchtb.com/wp-content/uploads/2011/01/image0070.gif)

在理解了上述HStore的基本原理后，还必须了解一下HLog的功能，因为上述的HStore在系统正常工作的前提下是没有问题的，但是在分布式 系统环境中，无法避免系统出错或者宕机，因此一旦HRegionServer意外退出，MemStore中的内存数据将会丢失，这就需要引入HLog了。 每个HRegionServer中都有一个HLog对象，HLog是一个实现Write Ahead Log的类，在每次用户操作写入MemStore的同时，也会写一份数据到HLog文件中（HLog文件格式见后续），HLog文件定期会滚动出新的，并 删除旧的文件（已持久化到StoreFile中的数据）。当HRegionServer意外终止后，HMaster会通过Zookeeper感知 到，HMaster首先会处理遗留的 HLog文件，将其中不同Region的Log数据进行拆分，分别放到相应region的目录下，然后再将失效的region重新分配，领取 到这些region的HRegionServer在Load Region的过程中，会发现有历史HLog需要处理，因此会Replay HLog中的数据到MemStore中，然后flush到StoreFiles，完成数据恢复。

## HBase存储格式

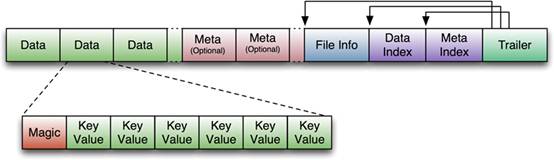
HBase中的所有数据文件都存储在Hadoop HDFS文件系统上，主要包括上述提出的两种文件类型：

1.       HFile， HBase中KeyValue数据的存储格式，HFile是Hadoop的二进制格式文件，实际上StoreFile就是对HFile做了轻量级包装，即StoreFile底层就是HFile

2.       HLog File，HBase中WAL（Write Ahead Log） 的存储格式，物理上是Hadoop的Sequence File

### HFile

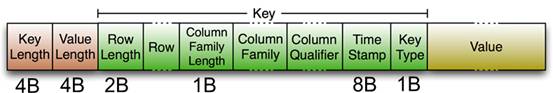
下图是HFile的存储格式：

[](http://www.searchtb.com/wp-content/uploads/2011/01/image0080.jpg)

首先HFile文件是不定长的，长度固定的只有其中的两块：Trailer和FileInfo。正如图中所示的，Trailer中有指针指向其他数 据块的起始点。File Info中记录了文件的一些Meta信息，例如：AVG\_KEY\_LEN, AVG\_VALUE\_LEN, LAST\_KEY, COMPARATOR, MAX\_SEQ\_ID\_KEY等。Data Index和Meta Index块记录了每个Data块和Meta块的起始点。

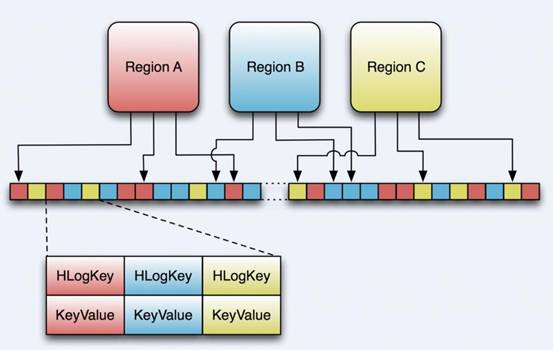
Data Block是HBase I/O的基本单元，为了提高效率，HRegionServer中有基于LRU的Block Cache机制。每个Data块的大小可以在创建一个Table的时候通过参数指定，大号的Block有利于顺序Scan，小号Block利于随机查询。 每个Data块除了开头的Magic以外就是一个个KeyValue对拼接而成, Magic内容就是一些随机数字，目的是防止数据损坏。后面会详细介绍每个KeyValue对的内部构造。

HFile里面的每个KeyValue对就是一个简单的byte数组。但是这个byte数组里面包含了很多项，并且有固定的结构。我们来看看里面的具体结构：

[](http://www.searchtb.com/wp-content/uploads/2011/01/image0090.jpg)

开始是两个固定长度的数值，分别表示Key的长度和Value的长度。紧接着是Key，开始是固定长度的数值，表示RowKey的长度，紧接着是 RowKey，然后是固定长度的数值，表示Family的长度，然后是Family，接着是Qualifier，然后是两个固定长度的数值，表示Time Stamp和Key Type（Put/Delete）。Value部分没有这么复杂的结构，就是纯粹的二进制数据了。

### HLogFile

[](http://www.searchtb.com/wp-content/uploads/2011/01/image0100.jpg)

上图中示意了HLog文件的结构，其实HLog文件就是一个普通的Hadoop Sequence File，Sequence File 的Key是HLogKey对象，HLogKey中记录了写入数据的归属信息，除了table和region名字外，同时还包括 sequence number和timestamp，timestamp是“写入时间”，sequence number的起始值为0，或者是最近一次存入文件系统中sequence number。

HLog Sequece File的Value是HBase的KeyValue对象，即对应HFile中的KeyValue，可参见上文描述。

# 结束

本文对HBase技术在功能和设计上进行了大致的介绍，由于篇幅有限，本文没有过多深入地描述HBase的一些细节技术。目前一淘的存储系统就是基于HBase技术搭建的，后续将介绍“一淘分布式存储系统”，通过实际案例来更多的介绍HBase应用。

[**分布式K/V存储方案 Cassandra**](http://www.oschina.net/p/cassandra)

**Apache Cassandra**是一套**开源分布式Key-Value存储系统**。它最初由Facebook开发，用于储存特别大的数据。**Facebook目前在使用此系统**。

主要特性：

* 分布式
* 基于column的结构化
* 高伸展性

Cassandra的主要特点就是它不是一个数据库，而是由一堆数据库节点共同构成的一个分布式网络服务，对Cassandra 的一个写操作，会被复制到其他节点上去，对Cassandra的读操作，也会被路由到某个节点上面去读取。对于一个Cassandra群集来说，扩展性能 是比较简单的事情，只管在群集里面添加节点就可以了。

Cassandra是一个混合型的非关系的数据库，类似于Google的BigTable。其主要功能比 [Dynomite](http://www.oschina.net/p/dynomite)（分布式的Key-Value存 储系统）更丰富，但支持度却不如文档存储[MongoDB](http://www.oschina.net/p/mongodb)（介于关系数据库和非关系数据库之间的开源产品，是非关系数据库当中功能最丰富，最像关系数据库 的。支持的数据结构非常松散，是类似json的bjson格式，因此可以存储比较复杂的数据类型。）Cassandra最初由Facebook开发，后转变成了开源项目。它是一个网络社交云计算方面理想的数据库。以Amazon专有的完全分布式的Dynamo为基础，结合了Google BigTable基于列族（Column Family）的数据模型。P2P去中心化的存储。很多方面都可以称之为Dynamo 2.0。

和其他数据库比较，有几个突出特点：

**模式灵活** ：使用Cassandra，像文档存储，你不必提前解决记录中的字段。你可以在系统运行时随意的添加或移除字段。这是一个惊人的效率提升，特别是在大型部 署上。   
**真正的可扩展性** ：Cassandra是纯粹意义上的水平扩展。为给集群添加更多容量，可以指向另一台电脑。你不必重启任何进程，改变应用查询，或手动迁移任何数据。   
**多数据中心识别** ：你可以调整你的节点布局来避免某一个数据中心起火，一个备用的数据中心将至少有每条记录的完全复制。

一些使Cassandra提高竞争力的其他功能：

**范围查询** ：如果你不喜欢全部的键值查询，则可以设置键的范围来查询。   
**列表数据结构** ：在混合模式可以将超级列添加到5维。对于每个用户的索引，这是非常方便的。   
**分布式写操作** ：有可以在任何地方任何时间集中读或写任何数据。并且不会有任何单点失败。

**Cassandra最新更新资讯，共17条  (**[**投递新闻**](http://www.oschina.net/home/go?page=admin%2Fnew-release)**，**[**查看所有»**](http://www.oschina.net/p/cassandra/news#list)**)**

* 3个月前[Cassandra 1.0 稳定版发布，NoSQL服务器](http://www.oschina.net/news/22393/cassandra-1-0-stabled) 2评/1547阅
* 4个月前[NoSQL大热 DataStax融资1100万](http://www.oschina.net/news/21744/nosql-datastax) 6评/1828阅
* 7个月前[Apache Cassandra 0.8.0 稳定版发布](http://www.oschina.net/news/18827/apache-cassandra-080) 0评/996阅
* 11个月前[Apache Cassandra 0.7.2 发布](http://www.oschina.net/news/15784/Apache-Cassandra-072-released) 1评/965阅
* 1年前[Apache Cassandra 0.7.0 正式版发布](http://www.oschina.net/news/14642/apache-Cassandra-070-released) 2评/1033阅
* **授权协议：**[Apache](http://www.oschina.net/question/12_2828)
* **开发语言：**[Java](http://www.oschina.net/project/lang/19/java)[查看源码»](http://www.oschina.net/code/explore/cassandra-0.7.0-rc2)
* **操作系统： 跨平台**
* **收录时间： 2008年10月06日**

# NoSQL数据库选型 DBA应该考虑什么？

我们曾经讨论过“到底NoSQL能在我们的工作中发挥什么作用?”我们也在考虑如何选择一款NoSQL数据库方面提出过101个相关问题。我们甚至召开了一个在线研讨会，深入剖析了SQL、NoSQL或者同时应用两者在网页应用程序的扩展性方面能带来哪些助益。

　　现在我们改变目标，转而思索哪些具体应用因素会对选择产生影响以及哪种系统在应对此类因素时更加适用。

　　你有什么意见?

　　首先，我们先来聊聊各类数据模型。下列相关信息参考自Emil Eifrem的博文及NoSQL数据库说明。

**文档类数据库**

　　·传承:受Lotus Notes启发而来。

　　·数据模型:文档汇总，包括键-值汇总。

　　·实例: CouchDB, MongoDB

　　·优势: 数据建模自然、程序员易于上手、开发流程短、兼容网页模式、便于达成CRUD(即添加、查询、更新及删除的简称)。

**图形类数据库**

　　·传承:来自 Euler 及图形理论。

　　·数据模型:节点及关系，二者结合能够保持键-值间的成对状态

　　·实例: AllegroGraph, InfoGrid, Neo4j

　　·优势:轻松玩转复杂的图形问题、处理速度快

**关系类数据库**

　　·传承:源自 E. F. Codd在大型共享数据库中所提出的数据关系模型理论

　　·数据模型:以关系组为基础

　　·实例: VoltDB, Clustrix, MySQL

　　·优势:性能强大、联机事务处理系统扩展性好、支持SQL访问、视图直观、擅长处理交易关系、与程序员间的交互效果优异

**面向对象类数据库**

　　·传承:源自图形数据库方面的研究成果

　　·数据模型: 对象

　　·实例: Objectivity, Gemstone

　　·优势:擅长处理复杂的对象模型、快速的键-值访问及键-功能访问并且兼具图形数据库的各类功能

**键-值存储**

　　·传承: Amazon Dynamo中的paper概念及分布式hash表

　　·数据模型:对成对键-值的全局化汇总

　　·实例: Membase, Riak

　　·优势:尺寸掌控得当、擅长处理持续的小规模读写需求、速度快、程序员易于上手

**BigTable Clones**

　　·传承自:谷歌BigTable中的paper概念

　　·数据模型:纵列群，即在某个表格模型中，每行在理论上至少可以有一套单独的纵列配置

　　·实例: HBase, Hypertable, Cassandra

　　·优势:尺寸掌控得当、擅长应对大规模写入负载、可用性高、支持多数据中心、支持映射简化

**数据结构类服务**

　　·传承: 不明

　　·实例: Redis

　　·数据模型: 执行过程基于索引、列表、集合及字符串值

　　·优势:为数据库应用引入前所未有的新鲜血液

**网格类数据库**

　　·传承:源自数据网格及元组空间研究

　　·数据模型:基于空间的构架

　　·实例: GigaSpaces, Coherence

　　·优势:优良的性能表现及上佳的交易处理扩展性

# Oracle 发布 NoSQL 数据库

Oracle 作为全球最大的关系型数据库提供商，在其产品链条中，也加入了NoSQL数据库这一环，而且这个新的数据库名字很霸气，就叫NoSQL Database，想起了当年新浪微博更换weibo.com 域名之时的一个笑话：

原来有三家人做面包，张三家的面包叫三张牌面包，李四家的牌子叫李四牌面包，王五家出品的是王五牌面包，而突然有一天，张三家的面包改名了，叫面包牌面包。李四王五异口同声的说：我擦！



Oracle的这个NoSQL Database，是在10月4号的甲骨文全球大全上发布的Big Data Appliance的其中一个组件，Big Data Appliance是一个集成了Hadoop、NoSQL Database、Oracle数据库Hadoop适配器、Oracle数据库Hadoop装载器及R语言的系统。

下面是NoSQL Database的特性简介：

* 数据模型简单
  + Key-Value式的存储，其中key由一级主要key和二次次要key组成
  + 由Java写成，支持基于Java API 的Put、Delete和GET操作
* 扩展性强
  + 支持自动地基于hash函数的数据分片策略
  + 提供基于数据节点拓扑结构和访问延迟的智能控制，以提供最佳的数据访问性能
* 行为可预测性
  + 提供ACID 的事务性支持，并且支持基于全局和单个操作的事务级别设置
  + 通过B-tree 数据结构构成的Cache层和高效的查询调度机制，提供可控的请求延时
* 高可用性
  + 没有单点故障
  + 提供内置且可配置的数据复制备份机制
  + 对单点或多点故障有很好的容错性
  + 通过跨数据中心的数据备份，提供数据的灾难恢复
* 简单的管理与维护
  + 除了命令行之外，还提供基于Web的界面管理工具
  + 提供对系统及数据节点的控制控制
  + 可以查看系统的拓扑结构、系统状态参数、当前负载情况、请求延迟记录、内部事件及通知等信息

这个Oracle NoSQL Database白皮书里还有更详细的讲解：[nosql-database-498041.pdf](http://www.oracle.com/technetwork/database/nosqldb/learnmore/nosql-database-498041.pdf)