非关系型数据库
HBase & NoSQL

授课老师：令狐东邪

课程版本： v1.0

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

本次课程大纲

1. 非关系型和数据库 NoSQL 简介

2. 使用 NoSQL 的场景

3. 为什么需要 HBase

4. 范围查询 Range Query

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

什么是非关系型和数据库 NoSQL

非关系型数据库可以简单的理解为存储在硬盘上的哈希表（HashMap）

常用的非关系型数据库大致分为这么几类

1. Key-Value 类

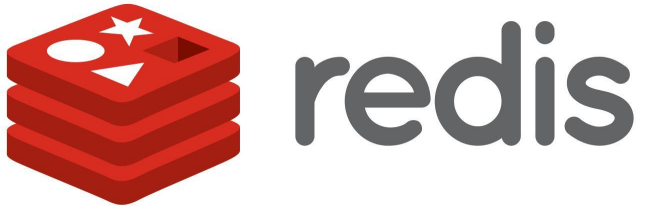
   Redis, RocksDB

2. Key-Document 类 (Document Based)

   MongoDB

3. Key-Key-Value 类 (Column Based)

   HBase, BigTable, Cassandra, DynamoDB



版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

为什么需要 NoSQL?

试想一种场景，我们需要在系统中记录用户的各种行为，方便之后通过用户行为复盘 BUG，数据格式大致为：

- 谁在什么时间干了什么事情
- user_id, timestamp, event_type, event_data

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

为什么需要 NoSQL?

请问这类数据有什么特点?

- 数据结构简单
- 数据量极大
- 读少写多

使用 MySQL 有什么地方不合适?

- 无法自动完成数据的分布式存储
- 写效率不够高

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

使用 NoSQL 的场景

数据结构简单，查询方式简单，写多读少，数据量大需要分布式存储

并非都需要满足，满足越多条件越适合使用 NoSQL

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

为什么需要 HBase

- Redis 也是 NoSQL，效率比 HBase 高，新版的 Redis 也支持了分布式，为什么不用 Redis 呢?
- 场景描述

  存储 NewsFeed，数据格式为 <user_id, timestamp, tweet_id>

  如果存储在 Redis 里，

  key = "newsfeeds::<user_id>", value = list(<timestamp,tweet_id>)

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

为什么需要 HBase

这样存储有什么问题?

- 不支持查询过去某段时间内属于某个 user 的 newsfeeds
- Redis 不支持范围查询（Range Query）

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

范围查询 Range Query

MySQL 中的范围查询

- CREATE INDEX newsfeeds_user_id_timestamp_index ON newsfeeds (user_id, timestamp)

- 有了这个索引之后，就可以支持如下两类查询
  1. 查询 user_id 在某个范围内的 newsfeeds（一般没有这个需求）
  2. 查询 user_id = <given user_id>, timestamp 在某个范围内的 newsfeeds（主要需求）

- MySQL 支持范围查询的原因：index 的结构是 B+ Tree

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

范围查询小练习

以下哪些场景和表单需要支持范围查询

A.Tweets 表单 (user_id, timestamp, content)

B. Friendships 表单 (from_user_id, to_user_id, timestamp)

C.UserActivities 表单 (user_id, timestamp, event_type, event_data)

D.Students 表单

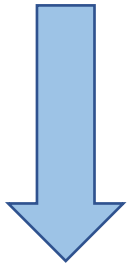
(student_id, class_id, department_id, name, email, phone, registered_timestamp)

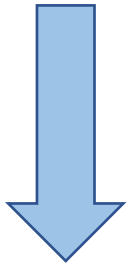
版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

HBase 的数据结构介绍

HBase 的每一条数据包含了三组信息

<row_key, row_data{column_key: column_value}>





key

value

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

HBase 和 MySQL 的存储结构对照

以 Tweet 为例子

HBase 的存储结构

row_key = {user_id} + {time stamp}	row_data = {
	'content': {content},
	'likes_count': {likes_count},
	'comments_count': {comments_count},
	}

"123+2021-11-11-11:11:11"	{"content": "hello", "likes_count": 0, "comments_count": 0}
"321+2022-12-12-12:12:12"	{"content": "world", "likes_count": 1, "comments_count": 1}

MySQL 的存储结构

primary key (id)	user_id	content	likes_count	comments_count
100	123	"hello"	0	0
200	321	"world"	1	1

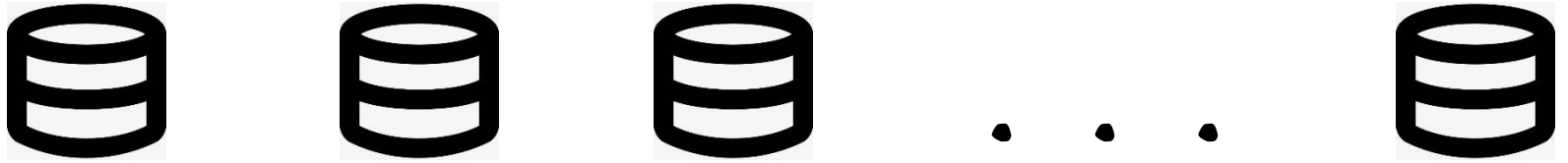
提问：那此时 HBase 对应 MySQL 的 Primary Key 是什么?

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

HBase 的 Range Query

HBase 的 Range Query 只针对 row_key 进行

HBase 的 row_key 从全局来看，是排好序的



当数据进行增删的时候，首先会根据 row_key 和每台机器管理的 row_key 范围找到这个数据应该存储的机器是哪一台，再到对应的机器上进行处理

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

HBase 的 Range Query

提问：在 HBase 中存储数据可否不带 row_key?

更多 HBase 的实现原理，请查看本周 Big Table 的那一节互动课

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

HBase 的 row_key 设计技巧一 - 翻转

HBase 的 row_key 如果没有设计好，会导致数据扎堆，从而导致在分布式系统中各台机器之间的存储容量或访问频次的热度不均匀，导致热点问题（Hotspot）

一个通用解决技巧是将 row_key 里如 user_id 或类似的和时序有关的，且不需要做范围查询的数据进行翻转

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

HBase 的 row_key 设计技巧一 - 翻转

比如以 NewsFeed 为例子，row_key 是 {user_id}+{timestamp}

比如 user_id=123，翻转之后得到 321。

原来的 key 是 "123+2021-11-11-11:11:11" 现在的 key 变为

"321+2021-11-11-11:11:11"

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

HBase 的 row_key 设计技巧二 - 补0

对于 row_key 中用于排序的属性，由于 row_key 的排序规则是字符串排序，因此如果这个属性本身的值是整数的话，需要进行补0操作

如整数 123，23，如果按照字符串排序，那么 123 会排在 23 的前面（即 123 更小），这显然不合理。因此如果按照 4 个字节的整数最多 10 位来补0的话，则应存储为 0000000123 和 0000000023，这样在排序之后 0000000023 会出现在 0000000123 的前面

具体补多少位 0 可以根据实际数据范围的情况来决定

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

HBase 的 row_key 设计技巧三 — 加盐

翻转的方法丢失了 row_key 中被翻转属性的有序性，是否有办法保留有序性有能够缓解热点问题呢?

我们可以在 row_key 的前方加盐（Salting），也就是增加一些固定长度的随机字符或数字。比如0到9或者a-z，又或者更长的随机前缀。

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任

HBase 的 row_key 设计技巧三 — 加盐

- 好处：能够一定程度的缓解热点问题
- 坏处：如果需要进行范围查询的时候，需要枚举所有可能的盐作为前缀分别查询，增加了查询的时间。

对于无需进行范围查询的数据，也可以直接在 row_key 中加上哈希前缀来避免热点问题。即

```
new_row_key = hash(old_row_key) + old_row_key
```

版权归属于九章算法（杭州）科技有限公司，贩卖和传播盗版将被追究刑事责任