### 背景

时序数据库可以看到是起监控作用的，如果你要监控很多东西的话，可以看出数据量会很大，并且每时每刻都在写入数据，并且还有数据的查询与分析，所以用mysql加上个时间戳列是玩不了的。

而早在2016年7月，百度云在其天工物联网平台上发布了国内首个多租户的分布式时序数据库产品TSDB，成为支持其发展制造，交通，能源，智慧城市等产业领域的核心产品，同时也成为百度战略发展产业物联网的标志性事件。时序数据库作为物联网方向一个非常重要的服务，业界的频频发声，正说明各家企业已经迫不及待的拥抱物联网时代的到来。

TSDB是百度云开发的一个时序数据库，时许数据库在物联网中会有很大发展。

百度无人车在运行时需要监控各种状态，包括坐标，速度，方向，温度，湿度等等，并且需要把每时每刻监控的数据记录下来，用来做大数据分析。每辆车每天就会采集将近8T的数据。如果只是存储下来不查询也还好（虽然已经是不小的成本），但如果需要快速查询“今天下午两点在后厂村路，速度超过60km/h的无人车有哪些”这样的多纬度分组聚合查询，那么时序数据库会是一个很好的选择。

也就是说这种需要实时监控，记录连续的随着时间的数据的需求就需要这个时序数据库。

### 时序数据库

先来介绍什么是时序数据。时序数据是基于时间的一系列的数据。在有时间的坐标中将这些数据点连成线，往过去看可以做成多纬度报表，揭示其趋势性、规律性、异常性；往未来看可以做大数据分析，机器学习，实现预测和预警。

时序数据库就是存放时序数据的数据库，并且需要支持时序数据的快速写入、持久化、多纬度的聚合查询等基本功能。

       对比传统数据库仅仅记录了数据的当前值，时序数据库则记录了所有的历史数据。同时时序数据的查询也总是会带上时间作为过滤条件。

### 属性

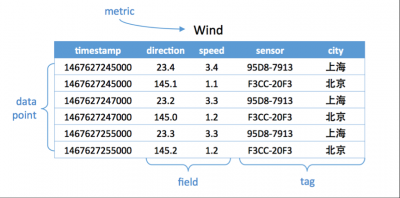
metric: 度量，相当于关系型数据库中的table。

data point: 数据点，相当于关系型数据库中的row。

timestamp：时间戳，代表数据点产生的时间。

field: 度量下的不同字段。比如位置这个度量具有经度和纬度两个field。一般情况下存放的是会随着时间戳的变化而变化的数据。

tag: 标签，或者附加信息。一般存放的是并不随着时间戳变化的属性信息。timestamp加上所有的tags可以认为是table的primary key。



度量为Wind，每一个数据点都具有一个timestamp，两个field：direction和speed，两个tag：sensor、city。它的第一行和第三行，存放的都是sensor号码为95D8-7913的设备，属性城市是上海。随着时间的变化，风向和风速都发生了改变，风向从23.4变成23.2；而风速从3.4变成了3.3

时序数据库可以看到是起监控作用的，如果你要监控很多东西的话，可以看出数据量会很大，并且每时每刻都在写入数据，并且还有数据的查询与分析，所以用mysql加上个时间戳列是玩不了的。

l   时序数据的写入：如何支持每秒钟上千万上亿数据点的写入。

l   时序数据的读取：又如何支持在秒级对上亿数据的分组聚合运算。

l   成本敏感：由海量数据存储带来的是成本问题。如何更低成本的存储这些数据，将成为时序数据库需要解决的重中之重。

### LSM tree

The Log-Structured Merge-Tree (LSM-Tree) 日志结构合并树。

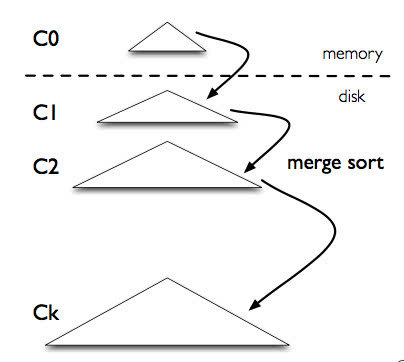
这玩意一般在时序数据库中才会使用，为啥呢，为啥不用B+树呢，B+树这种东西是查询快，写入慢，你还得维护索引数据。平时我们的业务主要是查询为主，所以用的B+树。

但是时序数据库这种东西明显是一直都在大流量的写入，只有在分析的时候踩死后查询。

LSM Tree是大流量写的时候用的东西。

LSM-tree 最大的特点就是写入速度快，主要利用了磁盘的顺序写，pk掉了需要随机写入的 B-tree。意思就是mysql如果你写的话如果这条数据存在的话,你需要找个这个数据并更新,但是对于lsmtree来说,直接在文件末尾嘎嘎的追加写就行,数据的更新是叫给文件合并的时候去做的.而且最新的一直在最上面

下图是 LSM-tree 的组成部分，是一个多层结构，就更一个树一样，上小下大。首先是内存的 C0 层，保存了所有最近写入的 （k，v），这个内存结构是有序的，并且可以随时原地更新，同时支持随时查询。剩下的 C1 到 Ck 层都在磁盘上，每一层都是一个在 key 上有序的结构。



写入流程：一个 put（k，v） 操作来了，首先追加到写前日志（Write Ahead Log，也就是真正写入之前记录的日志）中，接下来加到 C0 层。当 C0 层的数据达到一定大小，就把 C0 层 和 C1 层合并，类似归并排序，这个过程就是Compaction（合并）。合并出来的新的 new-C1 会顺序写磁盘，替换掉原来的 old-C1。当 C1 层达到一定大小，会继续和下层合并。合并之后所有旧文件都可以删掉，留下新的。

注意数据的写入可能重复，新版本需要覆盖老版本。什么叫新版本，我先写（a=1），再写（a=233），233 就是新版本了。假如 a 老版本已经到 Ck 层了，这时候 C0 层来了个新版本，这个时候不会去管底下的文件有没有老版本，老版本的清理是在合并的时候做的。

写入过程基本只用到了内存结构，Compaction 可以后台异步完成，不阻塞写入。

查询流程：在写入流程中可以看到，最新的数据在 C0 层，最老的数据在 Ck 层，所以查询也是先查 C0 层，如果没有要查的 k，再查 C1，逐层查。

一次查询可能需要多次单点查询，稍微慢一些。所以 LSM-tree 主要针对的场景是写密集、少量查询的场景。

LSM-tree 被用在各种键值数据库中，如 LevelDB，RocksDB，还有分布式行式存储数据库 Cassandra 也用了 LSM-tree 的存储架构。

### Tips

 可以看到各分布式时序数据库虽然存储方案都略有不同，但本质上是一致的，由于时序数据写多读少的场景，在单机上采用更加适合大吞吐量写入的单机存储结构，而在分布式方案上根据时序数据的特点来精心设计，目标就是设计的分片方案能方便时序数据的写入和读取，同时使数据分布更加均匀，尽量避免热点的产生。

       数据存储是时序数据库设计中很小的一块内容，但也能管中窥豹，看到时序数据库从设计之初就要考虑时序数据的特点。后续我们会从其他的角度进行讨论。