**我们为什么需要一个时序数据库？**

为了理解我们为什么需要时序数据库，我们首先需要来详细了解一下时序数据库以及其相关应用。

广为我们了解的所谓数据库，严格意义上来讲应该叫关系型数据库。关系型数据库的特点就是它在存储数据时采用表格的方式，数据将会被以行和列的方式进行存储，这种方式方便了读取和查询。由于关系型数据库具有命名规范化、数据一致且完整、冗余避免以及具有一个定义严谨的范式等清晰且便于理解的设计原则，它被运用在很多场合中，并且被证明是普适的。

但是关系型数据库在工业化生产中遇到了麻烦。在需要记录时间序列数据时，关系型数据库的各种操作就显得效率低下与繁琐。我们首先需要理解，在工业上，时间序列数据指的是由电力行业、化工行业、气象行业、地理信息等各类型实时监测、检查与分析设备所采集、产生的数据，它们的特点是产生频率快（每一个监测点一秒钟内可产生多条数据）、严重依赖于采集时间（每一条数据均要求对应唯一的时间）、测点多信息量大（常规的实时监测系统均有成千上万的监测点，监测点每秒钟都产生数据，每天产生几十GB的数据量）。那么直观上看，记录这类数据就需要进行大量的写入操作，并且需要数据库具有允许大规模改动的能力。根据相关专业知识我们可以简单分析出关系型数据库的劣势：夸张一点说，关系型数据库在如此之大的写入与改动要求面前就是静态的。

根据已经了解的关系型数据库设计原则，我们可以知道关系型数据库的属性以及属性容量在设计之初就已经决定好了，那么如果在之后的记录中数据出现突变，而属性容量又正好较小，这就会产生严重的后果，数据就不能被记录；如果所有的数据属性容量都被定义到很大，那么这会显得很浪费并且不专业。再者，关系型数据库是以表格的形式进行存储的，数据被存储在行和列中，这在方便了读取和查询的同时却使得写入操作变得困难，更不必说去改变数据。而刚好，时间序列数据就需要大量的写入以及改动操作。所以在面对时间序列数据的存储需求时，由于本身特性的限制，关系型数据库就显得力不从心，我们也急需一种能够有效存储时间序列数据的数据库出现。

接下来我们就要引出本文的重点，那就是时序数据库以及其相关特点的介绍。简单来说，时序数据库全称为时间序列数据库，时间序列数据库就是专门用于存储和处理时间序列数据的数据库。时序数据库具有一些基本的概念，这是和关系型数据库里面的行和列是不同的，二者需要进行区分：

1，度量：度量就类似于关系型数据库里面的表，代表一系列同类时序数据的集合。例如在存储某个温度传感器的监测数据时，我们就需要创建一个度量来存储这些时序数据；

2，标签：标签用来描述数据源的特征，通常不随时间变化。依然用温度传感器举例，传感器设备、设备所在地区的这些固定信息就被认为是标签。时序数据库内部会为标签建立索引，支持关于标签的相关检索操作。标签由标签键和标签值组成，二者都是String类型；

3，时间戳：时间戳代表数据产生的时间点，可以写入时指定，也可由系统自动生成；

4，量测值：量测值描述数据源的量测指标，通常随着时间不断变化，例如温度传感器就包括温度这个量测值；

5，数据点：数据源在某个时间产生的某个量测指标值称为一个数据点，数据库查询、写入时按数据点数来作为统计指标；

6，时间线：数据源的某一个指标随时间变化，形成时间线，度量+标签+ 量测值组合确定一条时间线；针对时序数据的计算包括降采样、聚合（sum、count、max、min等）、插值等都基于时间线维度进行.

时序数据库在操作中也和传统的关系型数据库有着显著的不同。首先我们可以考虑数据写入方面，在这方面根据时序数据的特性我们可以总结出如下的写入特点：

1.数据量大：拿监控数据来举例，如果我们采集的监控数据的时间间隔是1s，那一个监控项每天会产生86400个数据点，若有10000个监控项，则一天就会产生864000000个数据点。在物联网场景下，这个数字会更大。整个数据的规模，是TB甚至是PB级的；

2.冷热分明：时序数据有非常典型的冷热特征，越是历史的数据，被查询和分析的概率越低；

3.具有时效性：时序数据具有时效性，数据通常会有一个保存周期，超过这个保存周期的数据可以认为是失效的，可以被回收。一方面是因为越是历史的数据，可利用的价值越低；另一方面是为了节省存储成本，低价值的数据可以被清理；

4.多精度数据存储：在查询的特点里提到时序数据出于存储成本和查询效率的考虑，会需要一个多精度的查询，同样也需要一个多精度数据的存储。

由于时序数据库要实现以上特点，这就要求它必须具有某些特性，或者说是满足某些特性的能力。时序数据库并不是加上了时间戳的关系型数据库，它是需要在海量数据的场景下进行使用的，所以它必须具有和传统的关系型数据库不一样的特性来解决大量的时间序列数据问题。

首先时序数据库必须能够支持时序数据的写入，而这种写入通常都是每秒上千万上亿数据点类型的；同时时序数据库必须支持时序数据的读取。同样地，这种读取也是秒级上亿数据的分组聚合运算；除此之外，时序数据库必须解决成本敏感问题。这是指海量数据存储带来的成本问题。如何低成本存储海量数据，这将成为时序数据库需要解决问题之中的重中之重。

针对这些棘手的问题，不同的时序数据库有着不同的解决方案。有一种叫做RRD（Round Robin Database）的数据库，这种数据库是一个环形的数据库，数据库由一个固定大小的数据文件来存放数据。此数据库不会像传统数据库一样为随着数据的增多而文件的大小也在增加。RRD在创建好后其文件大小就固定，可以把它想像成一个圆，圆的众多直径把圆划分成一个个扇形，每个扇形就是可以存数据的槽位，每个槽位上被打上了一个时间戳。在圆心上有一个指针，随着时间的流逝，取回数据后，指针会负责把数据填充在相应的槽位上。当指针转了360度后，最开始的数据就会被覆盖，就这样RRD循环填充着数据。

另外一种解决方案是InfluxDB给出来的。InfluxDB自己造了一个叫做Time Structured Merge Tree的轮子，简称TSM。TSM的名字很有误导性，它并不是一种树结构，更不是日志，而是Merge。TSM在写入的时候，会把数据先写入到内存里，之后再批量写入到硬盘。在进行读取操作的时候，TSM会同时读内存和硬盘然后合并结果。在进行删除的时候，TSM写入一个删除标记，这样被标记的数据在读取时不会被返回；后台会把小的块合并成大的块，此时被标记删除的数据才真正被删除。相对于普通数据，有规律的时间序列数据在合并的过程中可以极大的提高压缩比。这样看来InfluxDB自己创造的TSM在时序数据库的应用方面的确具有很不错的效果。

目前看来时序数据库仍然属于新兴产业，很多公司也在尝试开发自己的时序数据库。在现在的互联网行业中，时序数据库拥有很高的热度，有不少的时序数据库可供选择使用。专门的时序数据库有TimescaleDB、KairosDB，以及上面提到过的InfluxDB等数据库。某些通用数据库也支持时序数据库，比如CrateDB和Kudu。由于篇幅限制，这里我就不会全部介绍。我会挑选TimescaleDB、CrateDB以及InfluxDB来进行简单的介绍。

TimescaleDB是基于PostgreSQL数据库打造的一款时序数据库，支持SQL。TimescaleDB拥有插件化的形式，并且会随着PostgreSQL的版本升级而升级，不会因为另立分支带来麻烦。由于本来的架构特点，TimescaleDB的数据会随着时间和空间进行自动分片。Timescale DB的分片同样能够进行自动的大小调整。总而言之，TimescaleDB是一种可以进行自动优化的时序数据库，提供丰富的SQL功能。可能TimescaleDB可以称得上不足的地方就是只适合数据量不太大的情况。

CrateDB基于Elastic Search，但它同样也支持SQL。CrateDB结合了SQL的通晓程度和NoSQL的可扩展性和数据灵活性，使得用户能够以实时速度使用SQL进行查询、JOIN和聚合以及进行简单缩放。CrateDB具有开放性、灵活性、简单可扩展性和高可用性的特点，并且可以进行备份。CrateDB适用于进行快速搜索，以及在写入数据的同时查询数据的情况。CrateDB支持大量数据，提供了高度可用的数据库。但是如果需要强事务一致性以及包括许多表和连接的高度规范化的模式时，CrateDB就不是一个很好的选择。另外，CrateDB的SQL支持度和读取性能会差一些。

InfluxDB是一个由InfluxData开发的开源时序型数据库，之所以介绍它是因为这种数据库在时序型数据库排行榜，也就是DB-Engines Ranking上排名第一。InfluxDB部署简单、使用方便，无需任何外部依赖即可独立部署。InfluxDB提供类似于SQL的查询语言，接口友好，具有丰富的聚合运算和采样能力。InfluxDB是针对时序存储、高性能读写、实时操作、高可用性而设计的一套软件。InfluxDB的各类性能非常强大，相比OpenTSDB、MongoDB、Graphite、Cassandra等，InfluxDB的性能优势和成本优势明显。除了以上特点，InfluxDB还具有无结构化的数据模型，丰富的权限和时效管理系统，以及免插件配置和第三方依赖的原生HTTP管理接口。InfluxDB的类SQL查询语言学习成本低、上手快。InfluxDB也有一些缺陷，例如并发查询性能较差等。但是在如此强大的性能和成本优势的前提之下，InfluxDB的瑕疵并不影响它的排名以及风评。

在花了大量篇幅对于时序数据库进行介绍之后，我相信我们对于开头的问题已经有了答案。不是说我们为什么需要时序数据库，而是说这是一种硬性的生产和研究需要。作为当前的互联网热点之一，时序数据库具有丰富的可开发性以及应用前景。我们对于时序数据库的需要更多的是出于对互联网技术的不断探索而产生的技术进步要求。