SQL vs NoSQL 对比之 Postgres vs Mongo

AI时间 2015-09-14

译者：施聪羽

原文链接：https://www.airpair.com/postgresql/posts/sql-vs-nosql-ko-postgres-vs-mongo

本文将对比最常使用的document数据库（Mongo）和最常用的关系数据库（Postgres）。我们将着眼于双方的优势和弱点，并尝试提供最切实可行的建议。当然，这绝不是数据库的全面调查。并不会讨论整个数据库家族（如图数据库）。这篇文章假设读者已经掌握了一定水平的技术能力。如了解JSON，这将是有利的。关于数据库，如果不太了解，并没有关系，本文会对背景理论进行说明。如果你已经对数据库理论有深刻的理解，可以跳过第一章，而不会漏掉信息。第二章，会在高抽象层次上，对比两个数据库提供的功能，并尝试得出对比结论。这轮对比将以一方明显优势获胜，我会宣布KO，但是如果双方并没有明显差距，那么结论就是，按需决策技术，并阐述一些理由。

1. 基础理论

在本章中，我们将讨论一下数据库的理论背景。这里所讨论的一切，将被假定为这篇文章的其余部分的背景知识。

1.1 CAP定理

任何正式的数据库讨论都将包括，至少引用过CAP定理。CAP定理认为网络共享数据系统可以有以下三个保证：一致性（在任何给定的时间点，每一个节点看到相同的数据，并且数据是最新的）可用性（每个节点都能处理并返回正确请求结果），分区容忍性（系统功能，不论网络分区）。这个定理进一步用来大致的划分数据库类型，数据库分为三个种类，CA，CP，和AP。代表着满足其中两个标准的数据库系统。CA系统显然是不可能的，因为即使是一个单一的网络客户端也需要引入系统分区。任何供应商销售一个CA系统是彻头彻尾的谎言。CP系统将在一个分区的存在的情况下，提供一致的数据。如果无法保证数据的一致性，他们将拒绝服务请求。AP系统将在一个分区的情况下，继续作出响应，但将不作任何数据一致性保证。

1.2 事务

数据库事务，是指作为单个逻辑工作单元执行的一系列操作。

1.2.1 ACID事务

ACID代表：原子性（事务中的所有操作，都是一个单一的“原子”的一部分，要么所有操作都成功，要么都失败。）一致性（无论是事务的开始还是结束，状态都是有效的，不违反任何数据库规则）隔离性（事务之间相互隔离，并且不能互相干扰）持久性（所有已完成的事务，对数据库所做的更改持续的保存在数据库中）。一致性在CAP理论中，实际上是一个严格的保证，并且通常被称为线性一致性。

1.3 线性一致性

CAP中的C真正的意义，通俗来说，保证的是，一旦操作完成的，其随后所有的操作中，事务无关或节点，都应该看到一致的，持久化后的操作结果，或者之后的状态。简单地说，如果我们已经写数据到变量x，任何或者说，所有的修改操作必须获取正确的x变量的值。

1.4 标准化

维基百科对标准化的定义为：“组织化的关系数据库中的属性和表，来减少数据冗余的过程。“有许多标准化策略，通常数据库如果是第三范式（3NF），那么就是标准化的。这意味着： 1. 每个属性只包含原子值。这明确地规定了不能在叶节点存储复杂的JSON或阵列结构。 2. 没有数据包含非唯一的子集。换而言之，对于每一个独特的条目集（这被称为候选关键字），没有其他的属性依赖于候选键的任意子集。 3. 没有数据是依赖于非主键。一些例子：让我们看下面简单的表格：

```
dob name attributes 1/1/1936 Edsger {hair:blue, eyes: green}
2/2/1937 Alan {hair: green, eyes: purple}
```

该张表将不满足第一个条件（通常被称为第一范式或1NF），因为属性列是非原子。现在让我们传递该标准，满足第一条件但是不满足第二条件（通常被称为第二范式或2NF）。

```
specialty name eye***color*** ***hair***color databases Edsger blue
greenprogramming Alan green purple algorithms Alan green purple
algorithms Edsgerblue green
```

这张表不满足条件，因为无论是specialty或是name都不足以唯一地标识一条记录，但是eye和hair color仅依赖name，从而导致信息的重复。最后，让我们来看看下表：

```
specialty name company***worked***at company_address databases Edsger
Foobar LTD1 Street programming Alan Foobar LTD 2 Street
algorithms> Alan Fubar GMBH 3Street algorithms Edsger Blue Corp 4
Street
```

这张表显示了哪些公司的专家在做什么工作，以及能力，但是地址应该与公司关联而不是与专家或者他们的specialty，这违反3NF。简单地说，这可以被铭记为“一切都必须与主键有关系，但事实上没主键什么事，帮帮我吧Codd”。（Codd是创造了这些范式的科学家。）现在，在这一点上非常合情合理的问题可能是“为什么这样做有用？”和“如何将这种设计付诸实践”。总之，通过标准化的数据，我们确保数据不重复，我们确保Postgres的查询优化器可以做出最好的决定（这个确切的算法不在本文的讨论范围之内），确保我们没有被破坏的数据，在系统中的任何位置（例如由于废弃的连接表）。

1.5 关系数据库(RDBMS)

一个关系数据库管理系统（RDBMS）是一种基于关系模型的数据库。它的目的是关注数据之间的关系，并存储在数据表中的行的列。一个关系模型背后的数学详细讨论超出了本文所涉及的范围，但是毫无疑问标准化的概念非常大程度上被应用到RDBMS。

1.6 文档数据库

文档数据库被设计用于存储所谓的半结构化数据，数据在有数据的模式之间没有明显的分离（思考一下数据的格式，有什么样的关系，多少列等），以及数据本身。

1.7 索引

维基百科这样定义一个数据库索引：

> 数据库索引是一种数据结构，该数据结构可以在一个数据库表中提高数据检索操作的速度，附加写入和存储空间的费用来维持索引数据结构。更笼统地说，数据库索引是一种数据库以跟踪特定列或属性的值的方法，从而使该列或属性更快速地读或搜索，但更慢的写入。

2. 对比

2.1 分类

2.1.1 Postgres分类

Postgres是ACID事务兼容的RDBMS，先进的，并包括事务完全串行化，如果事务级别被设置为可序列化。其查询语言是SQL，并且它是CP。

2.1.2 MongoDB分类

MongoDB是一个多存储引擎的野兽。默认存储引擎是MMAP，虽然WiredTiger是声称解决一些比较系统性缺陷，最新发布的存储引擎。然而大量的bugs显示WiredTiger有数据丢失和内存泄露的漏洞，它显然还需要进一步验证。因此，虽然WiredTiger声称将被改进，但是我们这里将使用MMAP引擎用于对比分析。MongoDB是一个无模式，面向文档的数据库管理系统，使用JSON等方法形成的模型对象。它并不直接的支持事务。它不是AP，因为它是一个单点master的系统，并且所有读都由主节点处理（尽管这可以通过改进使用副本集和自动故障转移，由此在网络分区上选取新的主节点）。如果主节点发生故障，它也不是CP了，一致性便无法保证。

2.2 一致性和事务

2.2.1 Postgres的事务和持久化

Postgres不需要读锁（除非事务级别设置为可序列化），因为每一个事务都有数据库的快照。不一致的读（也称为脏读），因此不可能。 Postgres有3个级别的事务隔离。有关深入的讨论，看看文档。

首先是读提交。这是默认的。这意味着，查询开始时，事务中的每个查询只能看到已经提交给数据库的数据。查询的执行过程中没有发生任何更改，但是在查询过程中被其他事务处理的并发更改将泄漏到事务中。

第二个级别是可重复读。此事务级别保证在事务执行过程中并发事务的任何更改都将是可见的。在效果上，将事务更改应用于其他事务的快照上。如果一个事务失败，由于不同的事务有修改的数据，该事务将由于并发异常而失败，你必须在应用程序层面上重试事务。

最后是可序列化的。此事务级别保证任何执行的事务将有完全相同的效果，因为它们是被一个接一个的执行。虽然这可能会出现在表面上和可重复读非常相似。

一个短的（实际）例子（来自上面的文档）是一个银行允许你透支你任何一个账户，只要在您所有帐户的累计总和不是负的。恶意攻击者可能会试图利用可重复的读取，从所有账户同时抽出大量的资金。由于可重复读取获取快照，每个单独的交易都会成功，导致银行的损失巨大。这是值得注意的，在这一点上，虽然已经做了大量的工作，以优化更高的事务水平，然而，更高的事务水平带来了一些性能开销。是否使用更高的一致性带来的性能成本，是应该一遍一遍的评估，使用基准测试，测试您的数据和操作。

2.2.2 MongoDB的事务和持久化

MongoDB中不直接支持事务。但它确实允许，通过配置WriteConcern（实际上和事务并不相关），达到写操作的持久可靠性，。默认的配置是Acknowledged，这可以保证在写操作已经达到了数据库，但不保证该数据实际上已写入磁盘。其他选项是： `Journaled`：写请求已被写入MongoDB的日志（队列中的操作，尚未保存到磁盘） `Majority`：写请求已经传播到大多数节点，并且被它们执行。

虽然`Majority`保证在没有网络分区的一致性，网络分区可能会导致不一致的数据，和/或数据丢失，即使是一个单一的文档范围内。虽然MongoDB提供$isolated操作符，通过一个写锁保持一致性，然而操作符在shard的集群模式下并不能用，而且写入操作的失败并不会回滚整个事务。MongoDB也提供了一个宽松的二阶段提交的实现。

粗略地说，这将要求您：创建一个事务，队列化包含所有更改操作。执行每一个更改，按顺序进行，跟踪成功更改。若有任何更改失败，回滚已更改的每一个更改，并取消该事务。标记事务完成

请记住，这意味着你需要对每个操作进行编码，或回滚操作，这是一个非常容易出错的问题。完整的解决方案超出了本文的范围。这也意味着：1）你的事务必须由客户端处理，而不是通过MongoDB本身 2）所有的shard没有全局锁，这意味着document在事务过程中也可以被修改。

2.2.3 WiredTiger引擎的事务

Wiredtiger声称提供ACID事务（我没有足够的计算能力来做这个测试，不幸的是尚未有第三方的测试结果）。根据Wiredtiger文档的描述，它提供了快照隔离的最高级别，相当于Postgres的`Read Committed`。

赢家：Postgres。如果WiredTiger已经能够使用，那么这个差距会缩小一些。

2.3 性能和非标准化数据

2.3.1 Postgres非标准化数据

Postgres支持4种用于存储非规范化数据的数据类型。 `hstore`：这是用于存储平键-值对。它在很大程度上只存在于遗留使用。 `json`：将json转换为字符串存储。它会对json做格式校验，并提供便利的操作符，但是它并没有真正的提供索引，这也在很大程度上只为遗留使用。`jsonb`：用于将json作为二进制形式保存，并显示为json，而不是作为一个单一的文本值，并允许任意属性做索引，从而提高查找速度。它仍然需要对整个json做更新操作，不过9.5版本将有更方便的更新嵌套路径下数据的方法。 `array`：用于保存其他一些数据类型的数组（text，number，等等）。要了解更多关于这些操作符，请参考文档。

2.3.2 Mongo非标准化数据

存储JSON是Mongo的长项。Mongo存储数据以二进制格式，称为BSON，这是（大约）只是一个二进制表示的JSON的超集。大约这么说，是因为Mongo数据类型不足的原因，而超集的原因是，Mongo直接支持二进制数据。

2.3.3 BSON VS JsonB

那么，如何我们进行对比？聪明的读者会注意到的第一件事，就是JSONB和BSON之间的相似性，两者在内部存储为二进制的JSON结构。它们有什么不同？第一点是，JSONB将输出完全符合标准的JSON，如所描述由RFC[9]，而BSON并没有。然而这是一把双刃剑。例如，JSONB不支持原生二进制类型，或日期类型。

2.3.4 性能对比

这里表明：Postgres的速度更快，使用更少的内存和磁盘，更高性能的存储和读取JSON。

赢家：Postgres。技术性胜利。Postgres更快，更少的内存，更符合标准，那么MongoDB，如果你需要一些BSON内在的类型检查，类型检查必须在一个标准化的方式进行，而不是由表列，或者如果你需要在一个JSON属性做复杂的访问更新，Mongo勉强的胜利。一旦Postgres 9.5上线，这就不是差距了。

2.4 复杂模型关系，访问模式，标准化数据

2.4.1 Postgres标准化数据

这是Postgres支持的。它允许您将数据关系编码到表中，使用外键来编码表之间的关系。它允许你join表之间的数据来自跨表边界的数据。例如：`sql CREATETABLE USERS( id SERIAL PRIMARY KEY, organization_id INTEGER, name TEXT );CREATE TABLE ORGANIZATIONS( id SERIAL PRIMARY KEY, name TEXT );`现在要查询话可以这么做：`SELECT USERS.* FROM USERS, ORGANIZATIONS WHERE USERS.organization_id =ORGANIZATIONS.id AND ORGANIZATIONS.name LIKE '%bar%';`或者`SELECTUSERS.* FROM USERS INNER JOIN ORGANIZATIONS ON USERS.organization_id =ORGANIZATIONS.id WHERE ORGANIZATIONS.name LIKE '%bar%';`

2.4.2 MongoDB标准化数据

一般来说MongoCollections对应Postgres的表，而MongoDocuments映射到Postgres的行。需要注意的是，MongoDB不支持join，迫使你直接查询嵌套关系，如果选择直接存储Key ID或DBref，或直接查询嵌套

的对象。存储一个嵌套的对象时如下：`db.createCollection('users'); db.users.insert({name: 'jack',organization: {name: 'foo corp'}});`

是一个简单的解决方案，这也是非标准化的，从业务逻辑角度来看，假定organization并不会变化。但是如果不是这样，或者你想标准化该数据？MongoDB允许两种方式来实现这一：

1）DBrefs允许您直接在文档中引用其他文档。然而，这将迫使每次查询时都会额外的去获取关联的文档，并应（如MongoDB文档）尽量避免。

2）`db.createCollection('users');db.createCollection('organizations'); db.organizations.insert({name: 'foocorp'}); db.organizations.insert({name: 'bar corp'}); var foo =db.organizations.find({name: 'foo corp'}); db.users.insert({name:"Jack", organization_id: foo});`你可以在应用层的逻辑中去提取organization_id，获取单独的去获取organization，join到数据中。需要注意的是，这将会绕过你建立或使用的任何事务逻辑，除非你的事务逻辑是在应用程序级处理。所以赢家：Postgres。

2.4.3 性能对比

MongoDB也提供了一个aggregation框架，让用户模拟关系型世界的join操作。一个很好的（虽然有些偏见的写作风格）的性能比较，**这里提供**。Postgres的连接和聚集比Mongo快三倍。此外，MongoDB聚合管道只能处理单一的Collection。这里的一个重要说明是，这个基准测试只适用于一台数据库。

2.5 可伸缩性

从根本上说，有2种类型的伸缩，水平和垂直。垂直伸缩意味着将资源添加到一个给定的机器上。更多的内存，更多的处理器内核。水平伸缩意味着多台机器运行您的数据库。

2.5.1 Postgres可伸缩性

只要你能保持垂直伸缩，Postgres的伸缩是微不足道的。机器配置越好，提高分配到Postgres的资源，并不是一直有效。迟早性能会达到天花板。Postgres水平伸缩是非常困难的，但是可行。有几个有效的策略来实现这个。核心方法是复制读取（master允许写，其他多个只读），和分片。分片是有多种方案的复杂话题，从应用层负载均衡，到数据库级的逻辑存储shardid作为主键的一部分，在Instagram **[13]**的分享中详细讨论了这个，但是超出了本文的范围了。

2.5.2 Mongo可伸缩性

MongoDB在技术层面上支持分片。当shard发生时，collection由`shard key`分区。Mongo的`query routers`可以正确的识别读取分片。要达到良好的分片，平衡shard大小，最佳实践可以在文献**[14]**发现 赢家：MongoDB。由于天生支持分片以及易用。

2.6 快速原型

假如，你有投资者掐着你的脖子，你还未完成昨天原型。你会选择的数据存储技术是什么？而Postgres似乎比Mongo更好，我们来看看使用Mongo的几个优势： 1. 由于存储的方式是schema-less，因此需求的更改不需要不断地迁移 2. 你不需要考虑你的数据模型，确保标准化。 3. 你不需要编写SQL作为查询语言，是JSON格式的，任何有JavaScript的经验的人对Mongo的查询语言都能很快上手。 4. 通常来说很多数据，并不重要，您的公司可以允许数据的丢失，因此由Postgres的强有力的保障是不必要的。

缺点：所有的数据都同样可能被丢失。如果您的公司的客户是企业，或处理财务数据，MongoDB是不可能的选项。而且，虽然MongoDB更容易扩展，但是Postgres也是可扩展的。

胜利：MongoDB，技术制胜。假设你还没有拥有Postgres的和/或数据库的专业知识，MongoDB的简单的查询接口以及方便的架构迁移/维护，更方便地快速原型，将更适合你。要知道，选择以上方案时要根据自己的应用场景，个人的小规模的数据损失确实是不相关的，否则，你最终将不得不放弃你的数据库，重写大部分应用程序。

3 总结

Postgres明确的获得了这场斗争的胜利。但是MongoDB任然有它适用的场景。适用于聚合大量的分布式数据，并真正的存储非标准化的数据，多数时候数据关系是不存在的，documents都很大，主要是非结构化的，而且几乎没有重叠，而数据丢失是无关紧要的（用于日志分析和缓存）。然而作为一个通用的数据库，Postgres显然在这个舞台上占主导地位的战斗机，如果是需要一些非标准化的数据，比如说用户的一组可选参数（眼睛的颜色，身高，体重，头发的颜色），Postgres的jsonb是绰绰有余。