16 | 为什么不建议你使用存储过程?

2020-09-14 王磊

分布式数据库30讲

进入课程 >

讲述：王磊

时长 14:29　大小 13.27M

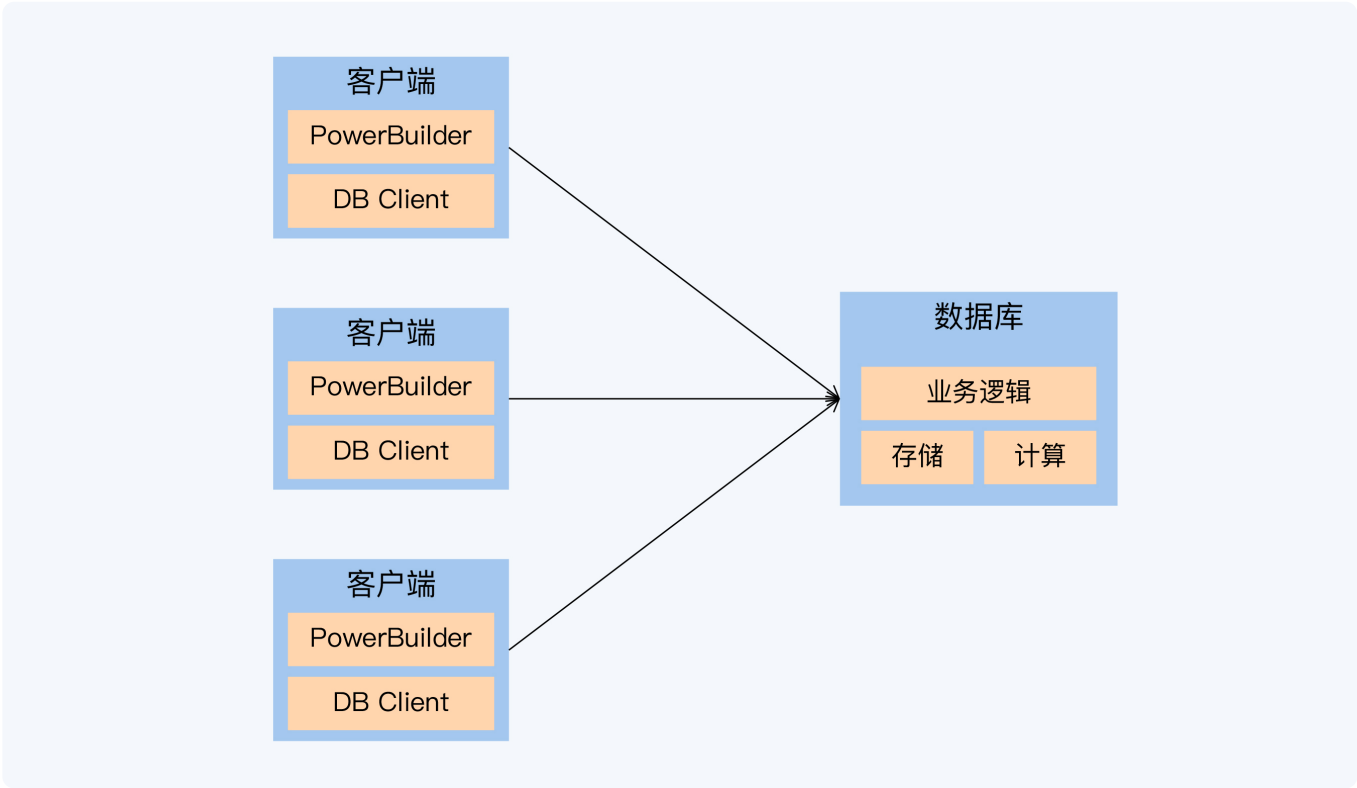
你好，我是王磊，你也可以叫我 Ivan。

今天，我们一起来到了这门课的第 16 讲。如果你学习并理解了前面的所有课程，那么我要恭喜你，这不仅是因为你学完了一半的课程，还意味着你已经征服了“数据库事务”这座高峰。当然，如果你有困惑的地方，也不必沮丧，因为接下来会是一小段平缓地带，我们会探讨一些相对独立的问题。比如我们今天的话题，为什么不建议你使用存储过程?

有些资深的数据库开发同学可能不同意这个观点，我猜他们大概会说：“存储过程很好呀，那些用不好的人就是自己水平烂，不接受反驳！”其实，我就有过这样的念头，但现在的我面对分布式数据库，会更倾向于少用或者不用存储过程。下面，我就来和你分享下这个心路历程吧。

我从 C/S 时代走来

当我刚成为一个名程序员时，正好是 C/S 架构时代的末期。那时最流行的开发套件是 PowerBuilder 和 Sybase 数据库。PowerBuilder 是一款可视化开发工具，有点像 VB，开发好的程序运行在用户的 PC 终端上，通过驱动程序连接远端的数据库。而 Sybase 当时正与 Oracle 争夺数据库的头把交椅，它和 SQL Server 有很深的渊源，两者在架构和语言上都很像。

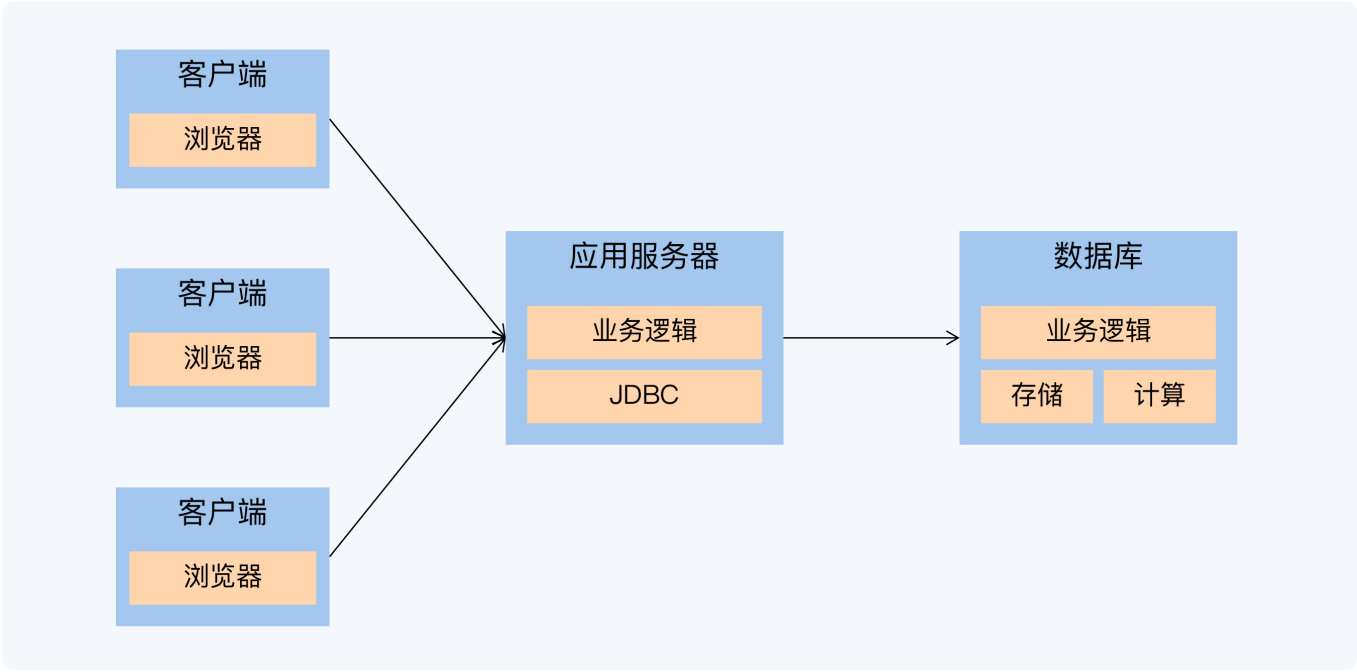


C/S架构

在这个 C/S 架构中，数据库不仅承担了数据存储、计算功能，还要运行很重的业务逻辑，相当于数据库同时承担了应用服务器（Application Server）的大多数功能。而这些业务逻辑的技术载体就是存储过程。所以，不管是 Sybase 还是 Oracle，它们存储过程的功能都非常强大。

触发器被抛弃

进入 B/S 时代，大家对数据库的理解发生了变化，应用服务器承载了服务器端的主要业务逻辑，那还要不要使用存储过程呢？你看，这和我们今天的问题是一样的。当时的主流观点认为存储过程还有存在价值的，但是它的同胞兄弟触发器则被彻底抛弃了。



B/S架构

为什么呢？其实，触发器和存储过程一样也是一种自定义函数。但它并不是显式调用，而是在操作数据表的时候被动触发，也就是执行 insert、update 和 delete 时；而且你还可以选择触发时机是在操作前还是操作后，也就 before 和 after 的语义。

听上去这个功能很强大吧，有点面向事件编程的意思。但是，如果你维护过触发器的逻辑就会发现，这是一个大坑。随着业务的发展和变更，触发器的逻辑会越来越复杂，就有人会在触发器的逻辑里操纵另一张表，而那张表上又有其他触发器牵连到其他表，这样慢慢就变成一个交错网络。

这简直就是一个地雷阵，你只要踏错一小步，经过一串连锁反应就会演变成一场大灾难。所以，触发器毫无悬念地退出了历史舞台。

存储过程的优点

存储过程的调用清晰，不存在触发器的问题。它的优点很明显，逻辑运行在数据库，没有网络传输数据的开销，所以在进行数据密集型操作时，性能优势很突出。

关于存储过程的使用，我有一段亲身经历，虽然过去了很多年但依然记忆深刻。当时要开发一个功能，追溯业务实体间的影响关系，比如 A 影响 B，B 又影响到 C。这个功能就是要以 A 为输入，把 B 和 C 都找出来，当然这个影响关系不只是三层了，一直要追溯到所有被影响的实体。

今天，我们都知道这是一个典型的关联关系查询，适合用图数据库来处理。但那个时候还没有可用的图数据库，我们需要在 Oracle 上解决这个问题。有一个比我更年轻的同事写了一段 Java 代码来实现这个功能，我猜他没有经历过 C/S 时代。程序运行起来，应用服务器不断地访问这张表，处理每一条记录的关联关系。性能可想而知，在一个数据量较少的测试环境上，程序足足跑了三十分钟。这大大超出了用户的容忍范围，必须要优化。

关于解决方案，我想你也猜到了，我换成了存储过程来实现同样的逻辑，因为不需要网络传输，性能大幅度提升。最后，存储过程花了大概二十几秒就得到了同样的结果。“干得漂亮！“我当时这么告诉自己。

存储过程的问题

但是后来，我发现了这个方案的问题，那就是移植性差。我们开发的产品要部署到客户环境里，会受到相关基础软件的制约。

有一次，刚好碰到这个客户没有使用 Oracle，所以其他同事将我写的逻辑翻写到了客户使用的数据库上。我们给这个数据库取个化名，就叫它 TDB 吧。可是，移植到 TDB 之后的存储过程并没有跑出结果，直接失败退出。我觉得很奇怪，就跟踪了这段代码，最后发现问题不在逻辑本身，而在数据库上。答案是这样的，这段逻辑中我使用了递归算法，因为 Oracle 支持很深的递归层次，所以运行完全没有问题；而 TDB 只支持非常有限的递归层次，而当时数据关联关系又比较多，所以程序没跑多久，就报错退出了。

这段经历让我对存储过程的信心有一点动摇。存储过程对于环境有很重的依赖，而这个环境并不是操作系统和 Java 虚拟机这样遵循统一标准、有大量技术资料的开放环境，而是数据库这个不那么标准的黑盒子。

然而，存储过程的问题还不止于此。当我在 C/S 架构下开发时，就遇到了存储过程难以调试的问题，只不过当时大家都认为这是必须付出的代价。但是随着 B/S 架构的到来，Java 代码的开发测试技术不断发展，相比之下存储过程难调试的问题就显得更突出了。而到了今天，敏捷开发日渐普及，DevOps 工具链迅速发展，而存储过程呢，还是“遗世独立”的样子。

说了这么多，我希望你明白的是，今天的存储过程和当年的触发器，本质上面临的是同样的问题：**一种技术必须要匹配同时代的工程化水平，与整个技术生态相融合，否则它就要退出绝大多数应用场景。**

你看，《阿里巴巴 Java 开发手册》中也赫然写着“禁止使用存储过程，存储过程难以调试和扩展，更没有移植性。”我想，他们大概是有和我类似的心路历程吧。

分布式数据库的支持情况

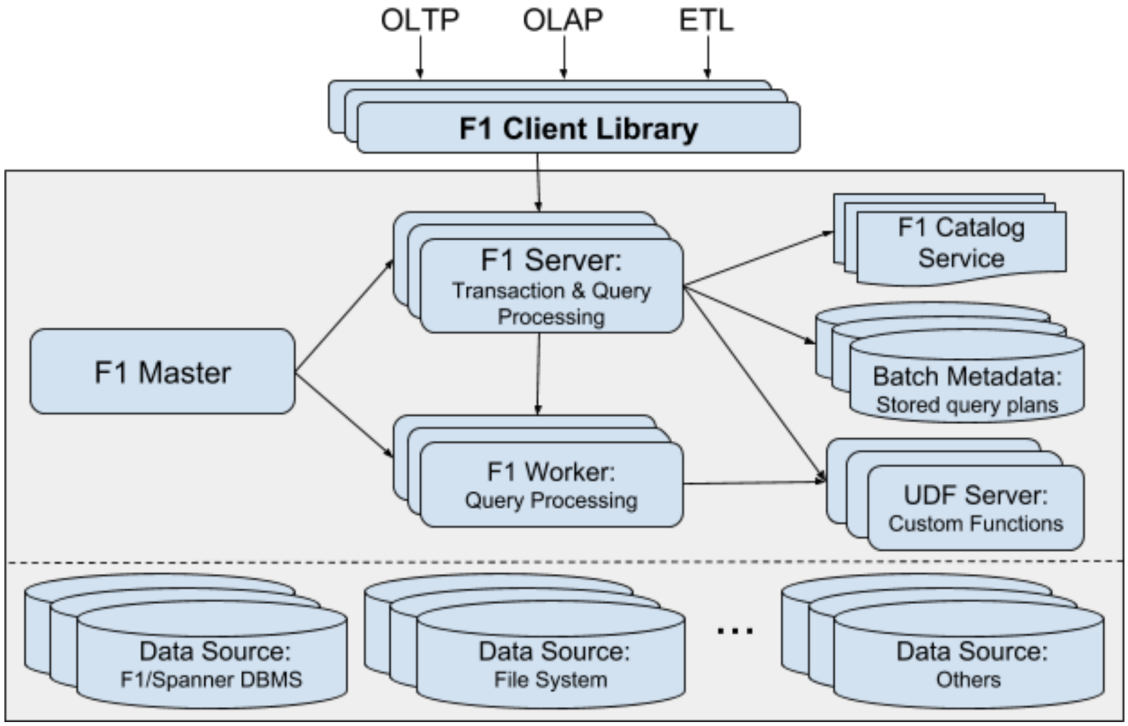
刚才说的都是我从工程化角度发表的一些观点，现在让我们回到分布式数据库，再来看看这个新技术对存储过程的支持情况是怎样的。

目前，多数 NewSQL 分布式数据库仍然是不支持存储过程的。OceanBase 是一个例外，它在 2.2 版本中增加了对 Oracle 存储过程的支持。我认为这是它全面兼容 Oracle 策略的产物。但是，OceanBase 的官方说明也说得很清楚，目前存储过程的功能还不能满足生产级的要求。

其实，对遗留系统的兼容，可能就是今天存储过程最大的意义。而对于那些从 MySQL 向分布式数据库迁移的系统，这个诉求可能就没那么强烈，因为这些系统没有那么倚重存储过程。其中的原因就是，MySQL 在较晚的版本才提供存储过程，而且功能上也没有 Oracle 那么强大，用户对它的依赖自然也就小了。

当然，存储过程没有得到 NewSQL 的广泛支持，还因为架构上存在的难题。我们不妨看看业界的一些尝试。

Google 在 2018 年 VLDB 上发布了 F1 的新论文”🔗F1 Query: Declarative Querying at Scale”。论文中提出，通过独立的 UDF Server 支持自定义函数，也就是存储过程。这个架构中，因为 F1 是完全独立于数据存储的，所以 UDF Server 自然也就被抽了出来。从论文提供的测试数据看，这个设计保持了比较高的性能，但我觉得这和 Google 强大的网络设施有很大关系，在普通企业网络条件下能否适用，这还很难说。



关于 UDF Server 的设计，还有两点也是非常重要的。

> 首先，UDF 实现了对通用语言的支持，除了 SQL，还支持 C++、Java、Go 等多种语言实现方式。这样不依赖于数据库的 SQL 方言，逻辑表述的通用性更好。
>
> 其次，UDF 并没有耦合在存储层。这意味着它的上下文环境可以更加开放。

这两点变化意味着存储过程的调试问题可能会得到明显的改善，使其与 DevOps 体系的对接成为可能。

不仅是 F1，其实更早的 VoltDB 也已经对存储过程进行了改革。VoltDB 是一款基于内存的分布式数据库，由数据库领域的传奇人物，迈克尔 · 斯通布雷克（Micheal Stonebraker）主导开发。VoltDB 将存储过程作为主要操作方式，并支持使用 Java 语言编写。开发者可以继承系统提供的父类（VoltProcedure）来开发自己的存储过程。下面是一个简单的示例。

复制代码

```
1 import org.voltdb.*;
2 public class LeastPopulated extends VoltProcedure {
3   //待执行的SQL语句
4   public final SQLStmt getLeast = new SQLStmt(
5       " SELECT TOP 1 county, abbreviation, population "
```

```
    + " FROM people, states WHERE people.state_num=?"
    + " AND people.state_num=states.state_num"
    + " ORDER BY population ASC;" );

  //执行入口
  public VoltTable[] run(int state_num)
      throws VoltAbortException {
          //赋输入参数
          voltQueueSQL( getLeast, state_num );
          //SQL执行函数
          return voltExecuteSQL();
      }
}
```

这段代码的逻辑非常简单，首先定义 SQL，其中“state_num=？”是预留参数位置，而后在入口函数 run() 中赋参并执行。

VoltDB 在设计理念上非常与众不同，很重视 CPU 的使用效率。他们对传统数据库进行了分析，认为普通数据库只有 12% 的 CPU 时间在做真正有意义的数据操作，所以它的很多设计都是围绕着充分利用 CPU 资源这个理念展开的。

具体来说，存储过程实质上是预定义的事务，没有人工交互过程，也就避免了相应的 CPU 等待。同时，因为存储过程的内容是预先可知的，所以能够尽早的将数据加载到内存中，这又进一步减少了网络和磁盘 I/O 带来的 CPU 等待。

正是由于存储过程和内存的使用，VoltDB 即使在单线程模型下也获得了很好的性能。反过来，单线程本身也让事务控制更加简单，避免了传统的锁管理的开销和 CPU 等待，提升了 VoltDB 的性能。

可以说，与其他数据库相比，存储过程对于 VoltDB 意义已经是截然不同了。

小结

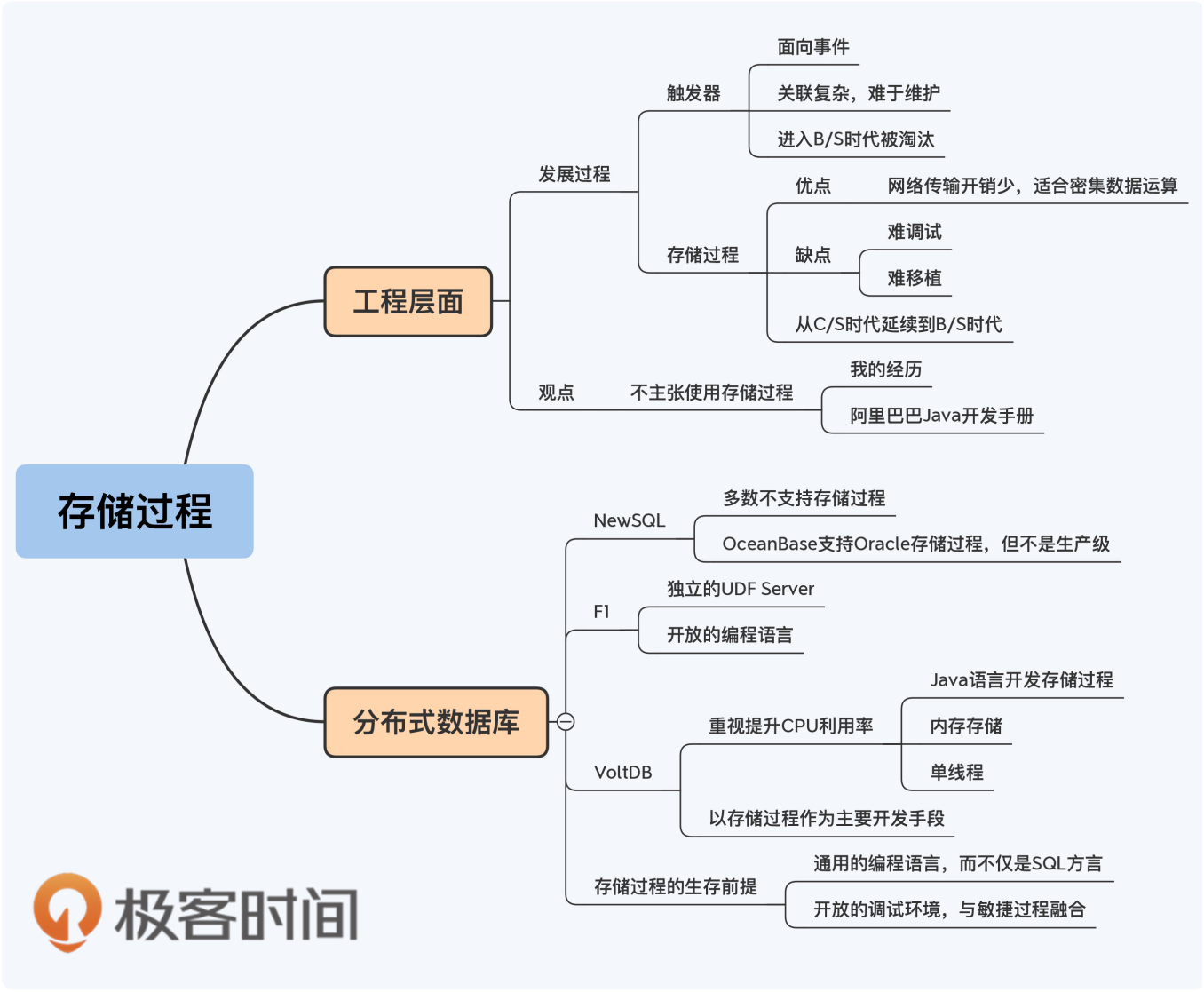
好了，有关存储过程的话题就到这里了，让我们一起梳理下今天的重点内容。

1. 我用自己的一段亲身经历，说明了存储过程的移植差。究其原因，在于存储过程高度依赖于数据库环境，而数据库环境不像操作系统或虚拟机那样遵循统一的标准。因为同样

的原因，存储过程调试也很复杂，也没有跟上敏捷开发的步伐，与今天工程化的要求不匹配。正是因为这两个工程化方面的原因，我建议你不用或者少用存储过程。

2. 从分布式数据库看，多数 NewSQL 还不支持存储过程，OceanBase 作为唯一的例外，已经支持 Oracle 存储过程，但仍然没有达到生产级。
3. F1 的论文提出了独立 UDF Server 的思路，是分布式架构下存储过程的一种实现方案，但能不能适合普通的企业网络环境，尚待观察。但这个方案中，存储过程的实现语言不局限于 SQL 方言，而是放宽到多种主流语言，向标准兼容，具备更好的开放性。这提升了存储过程技术与 DevOps 融合的可能性。
4. VoltDB 作为一款内存型分布式数据库，以存储过程作为主要的操作定义方式，支持使用 Java 语言开发。甚至可以说，VoltDB 的基础就是存储过程这种预定义事务方式。存储过程、内存存储、单线程三者互相影响，使得 VoltDB 具备出色的性能表现。

对于任何一个程序员来说，放弃一种已经熟练掌握而且执行高效的技术，必然是一个艰难的决定。但是今天，对于大型软件系统而言，工程化要求远比某项技术本身更加重要。不能与整个技术生态协作的技术，最终将无法避免被边缘化的命运。当你学习一门新技术前，无论是分布式数据库还是微服务，我都建议你要关注它与周边生态是否能够适配，因为符合潮流的技术有机会变得更好，而太过小众的技术则蕴藏了更大的不确定性。



思考题

课程的最后，我们来看看今天的思考题。我们说 VoltDB 的设计思路很特别，除了单线程、大量使用内存、存储过程支持 Java 语言外，它在数据的复制上的设计也是别出心裁，既不是 NewSQL 的 Paxos 协议也不是 PGXC 的主从复制，你能想到是如何设计的吗？提示一下，复制机制和存储过程是有一定关系的。

欢迎你在评论区留言和我一起讨论，我会在答疑篇和你继续讨论这个问题。如果你身边的朋友也对存储过程这个话题感兴趣，你也可以把今天这一讲分享给他，我们一起讨论。

学习资料

Bart Samwel: *F1 Query: Declarative Querying at Scale*

提建议

上一篇 15 | 分布式事务串讲：重难点回顾+思考题答疑+知识全景图

下一篇 17 | 为什么不建议你使用自增主键?

精选留言 (8)

写留言

Jxin
2020-09-16

对本章的理解:
1.业务代码跟技术代码应该要分离，我们需要保证业务逻辑到业务代码的翻译简单而纯粹。这既是在实现阶段，降低对具体技术的耦合，也是在保证业务代码的可测试性以及业务代码的简单和内聚。
2.具体技术的特性(比如存储过程)往往能起到杰出的性能。但这也增加了实现阶段的复杂...
展开

作者回复: 咋看字数很多，但每一条分析很清晰，总结的非常好，点赞。

2 2

佳佳的爸
2020-09-14

OceanBase支持Oracle的存储过程是迫不得已的事情，因为这决定着 它是否能 "侵入" Oracle的传统客户阵营-大企业和金融领域，是一种纯粹的商业行为。举个例子来说，Oracle的ERP产品中大量采用了存储过程来实现业务逻辑，最复杂的业务逻辑的源码打印出来几十页，这么复杂的存储过程 我相信 OceanBase的工具是无法完美处理(移植到OB)的，但是为了竞标之类的商业行为，你如果不支持Oracle的很多特性 你就根本没有参与的机会...

展开

1 2

佳佳的爸
2020-09-14

VoltDB用K-safety机制解决数据复制的问题，其实就是N+1的副本机制，VoltDB在写数据操作的时候，会在每个副本中执行该语句，这样就可以保证数据被正确插入每个副本。这N+1的副本都可以同时提供访问，同时允许最多N个副本丢失(分区故障), 当N+1个副本都不可用的时候，VoltDB就会停止服务进行修复。

展开

1

ifxdba
2020-09-23

存储过程就是一个死穴，一旦上路，便没有看了回头路

展开

作者回复: 也没有这么夸张啦:)，对一些特定需求还是有独特价值的。

vkingnew
2020-09-20

1.“《阿里巴巴 Java 开发手册》中也赫然写着“禁止使用存储过程，存储过程难以调试和扩展，更没有移植性。” 这个有误导的嫌疑，这里的存储过程针对的是MySQL，确实难以调试难以复制。在同等时代的产品产品中以oracle的PL/SQL,SQL server 的T-SQL编写的存储过程还是很有优势的，DB2和Oracle都支持PL/SQL的，这里PL/SQL是具有移植性

的，并且这个存储过程在SQL标准中叫做（SQL/PSM (SQL/Persistent Stored Module...

展开

作者回复: 我个人认为，阿里开发手册的这条建议不是特指MySQL，放在MySQL这一章可能因为MySQL在阿里使用比较多，是有代表性的数据库。难道他们一边宣传禁用MySQL存储过程，一边暗地里快乐的用着Oracle的存储过程？似乎不大可能。

另外，SQL标准对所有数据库都只是参照，不同的数据库，数据类型、全局变量、函数、甚至存储过程名的长度都有差异。没有完全相同的数据库，除非是专门适配。这也是为什说系统切换数据库是个大事。

了解这些差异后，有的同学可能依然觉得这不是事，so easy。对个体来说，难还是易是个很主观的判断，关键在于你的团队是否能长期、低成本的使用这项技术，如果可以那也未尝不可。

tt
2020-09-14

我觉得既然存储过程都支持用JAVA了，那数据复制应该就可以借鉴TCC，直接在代码层也就相当于是“服务层”实现，而且又是基于内存的，重试的成本还是比较低的，直接用代码往节点里写得了。

都是基于内存，但与REDIS不同，应该不会要求那么高的性能，直接用线程池同时往数...

展开

佳佳的爸
2020-09-14

存储过程是单机数据库时代的不可替代的产物，当年我当程序员的时候，存储过程是最好的解决前后端代码分离的利器。一个10万条订单批量审核的操作，调用存储过程几分钟搞定，前端vb代码执行，一个小时都出不了结果

展开

作者回复: 是的，存储过程对于数据密集型计算，绝对是一大利器。

地下城勇士
2020-09-14

clickhouse的物化视图其实是个触发器，这种应该从什么角度分析？