**摘要：**随着云数据库市场的不断增长，仔细检测和消除慢查询对服务的稳定性至关重要。以前的研究集中在优化由于内部原因（例如，编写不好的sql）而导致的慢查询。在这项工作中，我们发现了一组不同的慢查询，它们可能比其他慢查询对数据库用户更危险。我们将这种查询命名为间歇慢查询（iSQs），因为它们通常是由外部（例如，在数据库或计算机级别）的间歇性性能问题引起的。诊断ISQ的根本原因是一项艰巨但非常有价值的任务。

本文提出了iSQUAD，即间歇慢查询异常诊断器，该框架可以诊断iSQs的根本原因，并且不需要人为干预。由于问题的复杂度，出现了一个基于机器学习的算法，来描绘iSQs和根因之间的内在关系。但是面临着包括，多样性，标注开销以及可解释性等问题。针对这些挑战，我们设计了异常提取、依赖清理、面向类型的模式集成聚类（TOPIC）和贝叶斯案例模型四个部分。iSQUAD由离线聚类和解释阶段和在线根本原因诊断和更新阶段组成。dba只需要在离线阶段标记每个iSQ集群一次，除非在线阶段出现了一种新的iSQ。我们对来自阿里巴巴OLTP数据库的真实数据集的评估表明，iSQUAD实现了iSQ根本原因诊断的平均F1-分数为80.4%，在准确性和效率方面优于现有的诊断工具。

1. **介绍**

不断增长的云数据库服务，如Amazon关系数据库服务、Azure SQL数据库、Google云SQL和阿里OLTP数据库，是支持企业日常运营和业务的重要基础设施。数据库中的服务中断或性能中断会导致严重的收入损失和品牌损害。因此，数据库总是处于不断的监控之下，其中慢查询的检测和消除对服务的稳定性至关重要。大多数数据库系统，如MySQL、Oracle、sqlserver，都会自动记录完成时间超过用户定义的阈值[7,37,43]的详细查询信息，即slow queries。分析了导致查询效率低下的原因，如SQL语句编写速度慢等。然而，许多其他慢查询都是由外部（例如，在数据库或计算机级别）的间歇性性能问题造成的，我们将它们命名为间歇慢查询（iSQs）。

通常，iSQs是云数据库性能问题甚至失败的主要症状。由于iSQs是间歇性的，服务开发人员和客户希望它们能够像正常情况一样响应，而延迟的突然增加会产生巨大的影响。例如，在web浏览期间，iSQs可能会导致意外的网页加载延迟。据报道，每延迟0.1秒，亚马逊的销售额就会损失1%，而谷歌搜索结果每延迟0.5秒，流量就会下降20%。在一年的时间里，我们获得了几条阿里巴巴OLTP数据库DBA仔细记录的性能问题记录：当一个性能问题发生时，一个ISQ的爆发持续几分钟。事实上，手动诊断iSQs的根本原因需要几十分钟，这既耗时又容易出错。

在云计算中，诊断iSQs的根本原因变得至关重要和具有挑战性。首先，iSQs事件变得越来越普遍。为了更好的利用率，多个数据库实例可以驻留在同一个物理机器上，这反过来又会导致数据库间的资源争用。第二，iSQs的根源差异很大。云数据库的基础设施比内部数据库的基础设施更复杂[29]，这使得dba更难诊断根本原因。准确地说，这种复杂性可能是由实例迁移、扩展、存储去耦等引起的。第三，云中大量的数据库实例使得iSQs的数量非常庞大。例如，每天在阿里巴巴OLTP数据库中生成数万个iSQs。此外，预计到2020年，大约83%的企业工作负载将在云端[12]。这一趋势使得有效诊断iSQs的根本原因变得至关重要。

在这项工作中，我们的目标是诊断云数据库中iSQs的根本原因，只需最少的人工干预。我们从阿里巴巴OLTP数据库DBAs记录的故障记录中了解症状和根本原因，并强调以下四点观察结果：

1. DBAs需要扫描数百个关键性能指标（kpi），以找出性能问题的症状。DBAs将这些KPI分为八种类型，对应于不同的根本原因（如表1所示）。然而，传统的根源分析（RCA）[2，6，9，18]没有能力具体区分多种类型的KPI症状来诊断iSQs的根本原因。例如，通过使用系统监控数据，即单个KPI（或单一类型的KPI），我们通常无法确定iSQs的根本原因[10]。
2. KPI模式不同导致问题表现形式不同。的我们总结了三组对称的KPI模式，即spike up或down、level shift up或down和void。我们观察到，即使两个ISQ具有相同的异常KPI集（但具有不同的异常行为），它们的根本原因可能不同。因此，单纯地将KPI异常检测为正常或异常，我们无法准确地诊断iSQs的根本原因[6,45]。
3. 一个异常KPI通常伴随着另一个或多个异常KPI。某些KPI高度相关[24]，数据库中的快速故障传播使它们几乎同时异常。我们观察到KPI异常传播的方式可以是单向的，也可以是双向的。
4. 相似的症状与相同的根本原因有关。在每一类根本原因中，绩效问题的KPI症状是相似的，例如，同一类型的KPI可以相互替代，但异常类别保持不变。然而，列举和验证异常KPI与根本原因之间的所有可能的因果关系是不可行的[36]。

因此，具有不同KPI波动模式的iSQs似乎与不同的根本原因有着复杂的关系。为了发现和理清这种关系，我们努力探索基于机器学习（ML）的方法，但在此过程中遇到了许多挑战。首先，当iSQs发生时，需要正确检测异常KPI。传统的异常检测方法只识别异常本身，而不识别异常类型（即KPI波动变化，如尖峰上升或下降，水平移动或下降）。这些信息的可用性对于确保后续诊断的高准确性至关重要。其次，根据检测到的KPI波动模式，必须从候选数据中找出iSQ的根本原因。标准的监督学习方法不适用于此类诊断，因为根本原因的逐例标记是不可取的。一个iSQs可以触发许多异常的kpi，并导致大量的调查，需要dba花费数小时的劳动。第三，无监督学习是一个合适的方法来消除标注工作，它只有非常有限的效率。聚类方法很难保证直观（可解释性）和准确性。

为了解决上述问题，我们设计了iSQUAD（间歇慢查询异常诊断器），一个全面的iSQs根因诊断框架，对人为干预的要求较低。具体来说，我们采用异常提取和依赖清理来代替传统的异常检测方法来解决异常多样性的第一个挑战。为了减少标记开销，提出了一种面向类型的模式集成聚类（TOPIC），将具有相同根源的iSQs聚类到一起，同时考虑kpi和异常类型。通过这种方式，dba只需要在每个集群中探索一个具有代表性的根本原因，而不需要单独标记它们的数量。为了提高聚类的可解释性，我们利用贝叶斯案例模型为每个聚类抽取一个基于实例的表示，这样便于数据库管理员进行研究。简单地说，iSQUAD包括两个阶段：离线集群和解释阶段以及在线根因诊断和更新阶段。首先运行脱机阶段以获取群集和根本原因，然后联机阶段使用这些群集和根本原因进行将来的诊断。dba只需要为每个iSQ集群标记一次，除非出现新类型的iSQ。通过使用iSQUAD，我们大大减轻了云数据库平台上dba的iSQ根本原因诊断的负担。

我们工作的主要贡献如下：

我们发现了云数据库中间歇性慢查询的问题，并设计了一个称为iSQUAD的可伸缩框架，为iSQs提供准确有效的根本原因诊断。它采用机器学习技术，克服了在通用性、标记开销和可解释性等方面的固有障碍。

为了减少标记开销，提出了一种新的聚类算法。用KPI的异常提取代替异常检测来区分异常类型。

据我们所知，我们在数据库领域率先应用和集成了基于案例的贝叶斯案例模型，并将案例子空间表示引入数据库管理员进行标注。

我们对iSQUAD的评估进行了大量的实验，证明我们的方法的F1平均得分为80.4%，即比以前的方法高49.2%。

此外，我们已经在一个真实的云数据库服务中部署了iSQUAD的原型。iSQUAD帮助dba在80分钟内诊断出几百个iSQs的所有10个根本原因，这比传统的逐个诊断快大约30倍。

本文的其余部分安排如下：§2描述了ISQ其根本原因诊断的动机和挑战。§3概述我们的框架iSQUAD。§4讨论了iSQUAD中构建综合聚类模型的详细ML技术。§5展示了我们的实验结果。§6给出了一个真实世界云数据库的案例研究和我们未来的工作。§7回顾了相关工作，§8总结了论文。

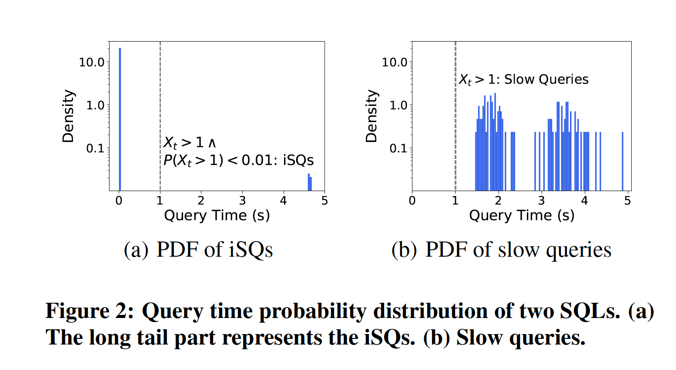
1. **背景和动机**

在本节中，我们首先介绍ISQ的背景知识。然后，我们从数据库性能问题记录中进行实证研究，以获得一些启示。最后，我们提出了诊断ISQ根本原因的三个关键挑战。

**2.1 背景**

**阿里巴巴OLTP数据库。**阿里巴巴OLTP数据库（简称阿里巴巴数据库）是一个多租户的DBPaaS，支持包括淘宝（customer to customer online retail service）、天猫（business to consumer online retail service）、鼎话（enterprise collaboration service）、菜鸟（logistics service）在内的多租户DBPaaS，这个数据库包含了超过10万个活跃运行的实例，跨越了几十个地理区域。为了监控对sla（服务等级协议）的遵守情况，数据库配备了一个度量系统[9]，该系统持续收集日志和kpi（关键性能指标）。

**间歇慢查询（ISQs）。**大多数数据库系统，如MySQL、Oracle、sqlserver，都会自动记录每次查询执行的查询时间[7,37,43]。查询时间是将SQL查询语句提交到数据库并返回结果给的时间。我们定义间歇慢查询（iSQs）如下：对于一个SQL查询语句Q，它的第t次出现Qt（观察到的执行时间为Xt）是一个iSQs**当且仅当**Xt>z且P（Xi>z）<，其中1 <=t；i <= t（t是Q最近出现的总数），z是慢查询阈值，是iSQ概率阈值。对于Alibaba数据库上的交互式事务性工作负载，dba根据经验设置z=1s，=0:01，T=104。请注意，这些阈值可以随着工作负载的变化而动态调整（例如，使用百分位和标准偏差），但是这并不是本文的重点。ISQ是间歇性发生的，由概率阈值来保证。例如，图2（a）示出了一个SQL的查询时间概率分布。在此图中，查询时间超过1秒的查询占用0.0028。这些iSQs是由间歇性的外部性能问题（例如，在数据库或机器级别）引起的。相反，图2（b）显示了另一个SQL，它是一个典型的慢查询，因为每次执行它都很慢。



表格

描述已自动生成

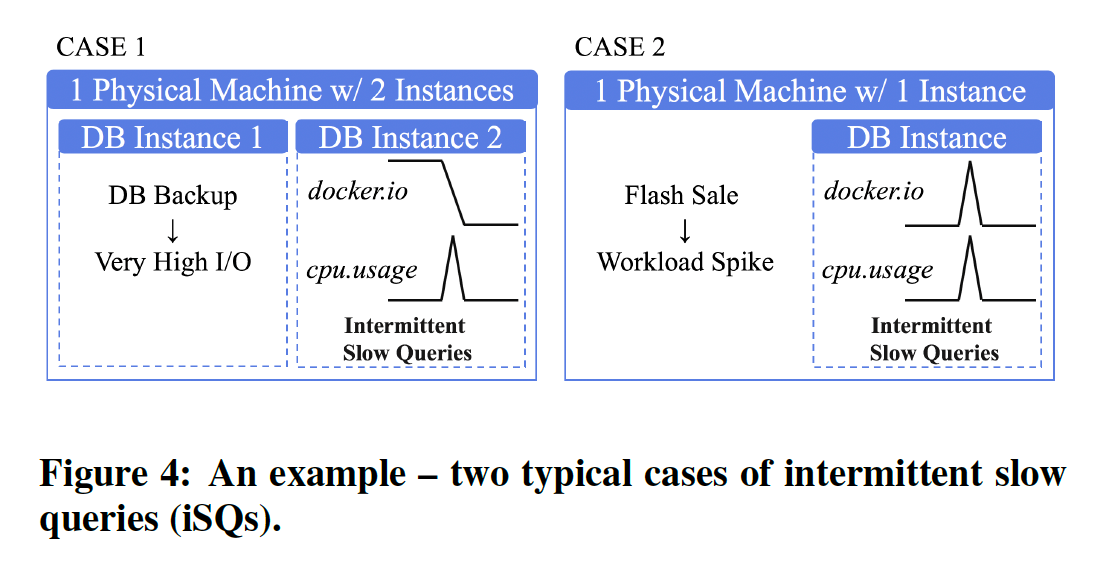
在这项工作中，我们关注iSQs的根因，这些根源可以通过kpi来解释或反映出来。而不仅仅是从物理配置中收集的，而且是从docker中收集的。对于每个iSQs，我们都会获得性能问题的确切时间和位置（实例或物理机器）。在经验丰富的DBAs的帮助下，我们选择了59个kpi，分为8种类型，如表1所示。它们涵盖了几乎所有可能导致ISQ的性能问题的特征。

**应注意kpi的异常类型。**性能问题症状可以用不同类型的KPI模式表示。从这些记录中，可以将KPI的表现归纳为四种异常类型，即峰值、水平上移、水平下移（KPI出现长时间的突然上升/下降或上升/下降）和无效（KPI值为零或缺失），如图3所示。以前的异常检测算法[31，33]关注kpi是否异常。然而，DBAs不仅检查异常的存在，而且更关注异常的确切类型。

图形用户界面, 应用程序, Word

描述已自动生成

我们提出了两个典型的iSQs案例。图4中的第一种情况，其中两个实例（实际上通常没有分配固定的I/O资源）同时在同一物理机器上运行。一个相关的数据库不可避免地会引发一个或多个相关的备份实例。由于这两个实例共享固定数量的I/O资源，实例2中的查询受到严重影响，因此看起来像是iSQ。这种情况表明，iSQs可能是由于其周围环境的负面影响而发生的，例如相关或“相邻”的慢查询。第二种情况涉及一台只有一个实例的物理机器。如果此实例的总体工作负载突然增加（例如，由在线闪购事件引起），则一个或多个与CPU相关的KPI可能会变得异常异常。因此，这个唯一实例中的查询将成为iSQs。第二种情况表明，异常工作负载也可能导致ISQ。



**KPI异常高度相关。**大多数情况下，一个异常KPI可能伴随着另一个或多个异常KPI。由于系统在组件之间具有复杂的关系，因此KPI之间高度相关[24]。我们发现故障传播可以是单向的，也可以是双向的，两个kpi之间的关系不一定是相互的。例如，实例上的异常(*docker.cpu-usage*）极有可能在物理机器上产生异常对应项(*cpu.usage*)，而物理计算机上存在问题的KPI(*cpu.usage*)可能并不总是在实例的KPI上看到相应的问题(*docker.cpu*-*usage*）。

**相似的KPI模式与相同的根因相关**。DBAs根据性能问题记录总结了云数据库中的十种根本原因（表2）。在每种类型的根本原因中，记录失败的KPI症状相似。同一类型的kpi可以相互替代，但它们的异常类型是恒定的。例如，“mysql.update-ps”和“mysql.delete-ps”在“mysql workload per second（ps）”的同一组中。它们都表示相同的根因:工作负载异常。因此，在执行RCA时，dba不必关心异常是由“update”还是“delete”的SQL引起的。

表格

描述已自动生成

云数据库具有实例迁移、容量扩展、主机资源共享或存储去耦等特性，这些特性可能会导致ISQ。我们解释了云数据库特性与根本原因的关系：在一个云数据库服务中，一台物理机可以托管多个数据库实例，这可能导致主机CPU、I/O、网络瓶颈等资源冲突。此外，密集的工作负载可能会更频繁地发生。例如，密集的工作负载是许多ISQ的根本原因。由于存储解耦，计算和存储节点之间的数据传输延迟并不总是很低[29]。因此，具有大I/O需求的查询可能会导致I/O瓶颈和随之而来的慢SQLs。

**2.3 挑战**

在将机器学习技术应用于我们的诊断框架时，我们遇到了三个挑战。

**异常多样性。**大量先进的异常检测器正在运行，并且一直在仔细检查KPI数据。它们中的大多数可以快速判断是否发生异常，但是这种二进制信息在我们的场景中是不够的。这是因为ISQ倾向于同时导致多个异常KPI，但事实上，这些KPI的时间线可能存在显著差异。在这种特殊情况下，仅区分正常和异常可能不会产生令人满意的结果，再次以图4为例。它们都包含相同的七个kpi，但异常类型不同。我们可能会得出错误的结论，即这两组性能问题（iSQs）具有相同的根本原因（而实际上并非如此）。此外，在检测大数据集中的异常时，优选具有高精度、低运行时间和高可扩展性的方法。

*现有解决方案的局限性：*异常类型的组合可能对应不同的根本原因，因此当前的异常检测器通常**忽略并过度概括异常类型**。这种检测器可能会错误地过滤掉（监控数据）预处理阶段的大量信息，从而降低（监控）数据集的质量。

**数据标记产生的间接费用**。由于可疑kpi的异常性能与其根本原因之间存在强烈的对应关系[6,45]，我们试图通过将DBAs的领域知识整合到我们的机器学习方法中来确定这种关系。为此，我们请有经验的DBAs标注iSQs的根因。然而，如果历史ISQs必须逐个手动诊断，则工作量是巨大的。即使DBAs拥有领域知识，标记过程仍然是痛苦的[31]。对于每个异常诊断，DBAs必须首先定位并登录到物理机器上，然后检查日志和与异常KPI相关的KPI，以得出诊断结论。为了成功地做到这一点，DBAs需要理解KPI功能和类别，找出异常KPI之间的联系，理解KPI组合，定位多个异常KPI、机器和实例，并预测可能的结果和对服务质量的影响。通常，DBAs会逐个分析异常，但这种诊断异常的方法既耗时又费力。例如，由有经验的DBA处理一个棘手的异常诊断案例可能需要几个小时甚至一整天。因此，仔细检查原始数据是乏味和容易出错的，而我们可以承受的错误容忍水平是非常低的。

*现有解决方案的局限性*：以前的工作[45]在试验台实验中再现根因，而不是标记根因。然而，在我们的例子中，简单地在一个试验台上复制已知的根本原因是不可行的，因为很难模拟如此大量的机器、实例、活动、交互等。此外，定制工作负载的数据集在可用性和维护方面通常都处于不好的状态。除了为实验复制原始场景的复杂性之外，即使我们设法重现过去的场景，实验统计数据也很难处理。

**可解释模型。**在我们的案例中，能够解释或叙述问题产生的原因（我们称之为可解释性）是至关重要的。为了做到这一点，需要向DBAs提供低于标准水平的机器和实例性能的具体证据，例如异常的kpi，以便他们能够采取相应的措施。DBAs通常不完全信任机器学习黑箱模型来为他们得出结论，因为这些模型往往产生难以概括的结果，而实时分析必须处理具有各种可能输入的不断变化的场景。因此，我们需要设计我们的诊断框架以获得更好的解释性。

不幸的是，在模型的准确性和它对人类的可解释性之间存在着不可避免的权衡[26]。这个问题的产生是因为系统复杂性的增加以牺牲可解释性为代价来提高其准确性，也就是说，当模型变得过于复杂时，人类很难理解结果和模型中的复杂性。因此，如何在我们的分析系统中同时达到良好的解释性和高精度，以及如何将贸易从边界向外推进，是一个具有挑战性的研究课题。

*现有解决方案的局限性*：使用决策树来解释模型是相当普遍的[15]。例如，DBSherlock[45]用类似决策树的实现构造基于谓词的异常说明。然而，可靠性在很大程度上取决于在一开始就提供精确的信息，因为即使输入中的细微差别也会导致大的树修改，这对准确性是不利的。此外，决策树还可能导致“分析瘫痪”的问题，即向决策者呈现的是过多的信息而不是关键元素。过多的信息会大大减慢决策过程并影响决策效率。

1. **概述**

我们设计了一个框架——iSQUAD（间歇慢查询异常诊断器），如图5所示。iSQUAD框架包括两个阶段：离线分析和解释以及在线根因诊断和更新。这种分离设计遵循离线学习和在线应用的共同模式。

图形用户界面, 应用程序, 日历

描述已自动生成

通常，具有相同或相似kpi的iSQs具有相同的根因。因此，有必要在模型中描述ISQ与其根因之间的联系。DBAs可以参与调查这种联系，因为他们的领域知识很高。在没有标记的情况下直接将根本原因分配给iSQ集群是不可行的。因此，离线阶段主要是基于标准对iSQs进行聚类，并将它们呈现给DBAs，DBAs可以更容易地识别和标记根本原因。我们将历史ISQs的数据集提供给离线阶段，然后集中在给定特定时间戳的特定时间间隔上。事情变得简单明了，因为我们可以只关注kpi时间表中选定的时间间隔，并在这些时间间隔内对kpi进行异常提取。接下来，我们将所有异常的kpi离散化。然后，我们对这个部分结果应用依赖清理。也就是说，如果我们有两个异常的kpi A和B，并且我们有领域知识A的异常倾向于触发B的异常，那么我们可以“清除”B上的异常警报，因此，我们可以假设在这一步之后，所有异常都是独立的。然后我们执行Type-Oriented Pattern Integration Clustering（TOPIC）以获得大量的集群。对于每个聚类，我们应用贝叶斯案例模型得到一个典型的iSQ及其基本KPI异常作为特征空间来表示整个聚类。最后，我们向DBAs展示这些集群及其表示，DBAs会调查iSQ集群并为其分配根本原因。

在线根本原因诊断和更新阶段中，iSQUAD自动分析传入的iSQ及其KPIs。我们像在离线阶段一样执行在线异常提取和依赖清理，得到异常kpi。随后，我们将查询匹配到一个集群。具体地说，我们根据相似度将这个查询与每个集群进行比较，然后将这个查询与模式最接近这个查询的集群进行匹配，然后，我们使用DBAs指出的这个集群的根因来帮助解释是什么触发了这个iSQ。如果查询与任何现有的集群都不匹配，那么将生成一个新的集群，DBAs将调查并为其分配一个根因。联机阶段的新发现可以更新脱机阶段的结果。

1. **iSQUAD详细设计**

在本节中，我们将介绍iSQUAD的详细信息，其组件与我们在§2.2中的观察结果相关。从第一次和第二次观察中获得见解，我们需要一种异常提取方法来从iSQ发生时的KPI统计中提取信息、数据、图形，以便准确地捕捉间歇性慢查询（§4.1.1）。根据第三个观察，我们必须消除KPI故障传播的影响，使 KPI的独立性得到保证。根据第四项观察，相似的症状与相同的根本原因相关。因此，我们提出了一种基于异常模式和KPI类型的查询聚类方法（§4.1.3）。由于缺乏特定案例的信息，聚类结果的可解释性不足以识别所有根因，因此采用贝叶斯案例模型（BCM）来提取聚类的“含义”（§4.1.4）。

**4.1离线分析和解释**

**4.1.1 异常抽取**

给定ISQ的发生时间戳，我们可以从数据仓库收集相关的KPI片段（如图1所示）。如前所述，我们需要从kpi中提取异常类型。例如，我们分别对应于图3中的（a）、（b）、（c）（d）部分，确定给定异常是峰值上升还是下降，电平上下移动，甚至是空洞。我们抓住这些宝贵的信息，因为它对于查询分类和解释非常有用

为了识别尖峰，我们应用了非常适合这种情况的鲁棒阈值[9]。作为平均值和标准差的组合来决定分布的一种替代方法，我们使用中值和中值绝对偏差的组合，它的工作更稳定，因为它不太容易出现数据混乱等不确定性。为了进一步抵消数据畸变的影响，鲁棒阈值利用柯西分布代替正态分布，因为前者在出现许多异常值时能更好地发挥作用。默认情况下，观察间隔设置为1小时，阈值是根据经验设置的。

请注意，还有其他各种优秀的异常检测器和算法，但是比较异常检测器并不是这项工作的贡献。从观测结果来看，这套异常提取方法既准确又实用。

**4.1.2 依赖清理**

为了更好地了解KPI对ISQ的影响，我们需要确保选择的所有KPI都是相互独立的，这样KPI的相关性或过度表达都不会影响我们的结果。对于每一个潜在的KPI，都需要一个潜在的比较。如前所述，两个KPI异常不一定具有相互关系。因此，与以往一些计算互信息进行比较的工作（例如DBSherlock）不同，我们使用基于两个kpi之间的关联规则学习的置信度[1]来确定两个kpi是否具有相关性。置信度表示if-then语句被发现为真的次数。

文本

描述已自动生成

其中A和B代表两个任意的kpi。具体地说，从A到B的置信度是A和B的异常同时出现的次数除以A的异常出现的次数。

置信值的范围从0到1，“0”表示两个KPI完全独立，“1”表示完全依赖。在这种情况下，不仅1表示依赖性。相反，在这个时间间隔内，我们设置了一个阈值，超过这个阈值，两个KPI被视为依赖于反映真实场景。我们对所有KPI进行排序，并将此策略应用于每个KPI配对。例如，一个实例的CPU利用率的异常通常伴随着实例的物理机器的异常。因此，这两个KPI在很大程度上是正相关的。如果我们计算置信度，我们可能得到结果“1”，这表明这两个KPI是相互依赖的。因此，我们删除了物理机CPU利用率的所有异常，并保留了实例的CPU利用率异常。在这一部分中，我们考虑异常传播来清理KPI异常，并保留源KPI异常。我们的依赖关系清理规则和结果由经验丰富的DBAs验证，如§5.4所示。

4.1.3 TOPIC

首先，我们让读者熟悉本节中使用的一些序言和术语。Pattern包含了iSQ的KPI状态（正常或异常类别之一）的特定组合。为了说明，图6中的两个查询具有两个相似但不同的模式。只要两者之间存在一个或多个差异，就认为两个模式是不同的。KPI类型（例如CPU相关KPI、I/O相关KPI）指示此KPI所属的类型。它包含一个或多个KPI，而KPI只属于一个KPI类型。我们可以根据KPI类型对KPI及其功能进行粗略分类（表1）。

表格

描述已自动生成

根据§2.2中的观察结果，我们需要同时考虑ISQ的模式和不同类型的KPI来计算相似度。我们将两个ISQ i和j的相似性Sij定义如下：

文本

描述已自动生成

其中t是KPI类型的数量，T表示所有t的总和。kit和kjt分别是iSQ i和j的KPI类型t中的KPI异常状态。此定义背后的思想是计算每种类型KPI的相似性得分的二次平均值。由于二次平均值不小于平均值，因此它保证最小的KPI更改不会导致将事件与另一个根本原因分组。｜kit；kjt｜是每种KPI类型的相似性，如等式3所示：

文本

描述已自动生成

这是简单的匹配系数[44]，它以位的方式计算两个元素的相似性。我们采用简单匹配系数，因为它反映了有多少个kpi具有相同的异常类型。然而，与数量较少的其他指标相比，某些类型的关键绩效指标中相对较多的指标可能占主导地位。例如，假设KPI类型“I/O”由18个KPI状态组成，而其对应的“CPU”只有2个（表1）。从理论上讲，“CPU”中的高相似度容易被“I/O”中的弱相似性所抵消。这种“平均主义”的方法不是我们所期望的。为了解决这个问题，我们决定根据不同的KPI类型来划分KPI，并为每个KPI类型计算单独的简单匹配系数。通过这样做，对于每种KPI类型，每对ISQ都将与区间[0；1]中的值具有“部分相似性”（与我们从所有KPI相似性的二次平均值中获得的“完全相似性”相反）。

我们描述了聚类过程的细节，如算法1所示。转换成字典的数据集包含isq及其通过异常提取和依赖关系清理离散化的模式。所需的输入（即阈值）用于确定两个ISQ的相似程度，以使其同质化。首先，我们将S反转为D：D的索引和值分别是S的值（模式）和聚集索引（isq）（算法1中的第2行到第3行）。对于全零模式，即KPI状态都是正常的，我们从D中消除它及其对应的isq，并将它们放入集群字典C（第4行到第6行）。这种先决条件检查保证了具有所有零模式的isq可以合理地聚集在一起。全零模式并不意味着完美无瑕。相反，它通常意味着MySQL内核有问题，这超出了本文的讨论范围。此检查的另一个目的是区分“X 0 0 0 0”和“0 0 0 0 0”的模式，其中X表示可以是任何类型的任意异常。前一种模式表示某个KPI在某种程度上是异常的，而后一种KPI完全安全可靠，显然它们在我们的场景中是不同的模式。然而，如果我们不在迭代之前消除D中的全零模式，这两个模式往往会聚集到同一组中。“Allzero模式”意味着kpi中没有异常。由于缺乏kpi异常症状，这些问题几乎无法诊断。我们关注可以由kpi解释或反映的根本原因，并将所有零模式问题留待将来的工作。

1. 评估
2. 分析
3. 相关工作

**慢速查询分析。**慢查询分析和优化已经得到了广泛的研究。一般的方法包括数据驱动的数据库和查询的自动分析和优化。对于数据库，以前的几项研究[13，28，40]旨在自动化索引修改以获得更好的性能，其中一项研究解决了使用机器学习算法调整数据库参数的问题[42]。对于查询优化，在[25，34]中引入了通过深度学习来提升查询的能力。以上两项都不涉及ISQ领域。我们的工作是第一次减少ISQ对数据库系统的负面影响。

**异常提取。**过去的异常检测算法通常输出二进制结果，即“正常”或“异常”。在文献中，有各种各样的异常检测器，如oprentice[31]、dSPOT[41]和iSST[33,46]。另外，一些公司开发了异常检测器，例如雅虎的EGADS[27]，Twitter的s-H-ESD[21]，Netflix的RPCA[17]。与它们不同，我们的异常提取返回KPI状态，即正常或以上介绍的异常类别之一，而不是有限的二进制结果。

**聚类算法。**一些与查询相关的聚类算法提供了见解。K-Shape聚类[38]建立在[5]之上，基于KPI时间线的形状对查询进行聚类。这种方法与我们的场景不同，因为我们关注所有kpi的一个时间戳，而K-Shape允许两个相似形状之间有一个时间差。这种延迟可能会将两个不相关的查询放在一起，并导致准确性损失。接下来，[32]中的工作负载压缩技术与我们的工作类似。基于余弦相似度计算工作负荷特征的相似度。一个缺点是它会丢失KPI类型的信息，而KPI类型对于确定查询行为至关重要。相比之下，我们考虑了KPI类型和异常模式，以严格的方式对查询进行聚类。此外，TOPIC并没有修改[32]等现有集群的聚类中心，即异常模式，因为在合并时集成的模式是稳定的，而不像[32]中随时间变化的模板，因此集群会更快地收敛。

1. 总结