分布式 数据库 系统设计

计算机科学 1994Vol. 21 №. 2

43-49

分布式数据库系统设计中的问题

TP311.13

郭 江

（北京航空航天大学软件工程研究所 北京 100083）

摘要 本文围绕着网络的规模、分布式的设计、查询处理、事务处理、支持分布式的操作系统、多数据库系统（MDBMS）等问题进行了探讨。如何理解和处理这些问题呢？本文从设计的角度出发，较为详细地讨论了这些问题对分布式数据库设计的影响以及设计时决策的常见选择方式。

关键词 多数据库系统 事务处理

现在，无论是商业数据处理，还是计算机工程环境（SEE）都迫切需要分布式数据库系统的支持。但是就目前的技术状况来说，关键的问题是怎样使分布式数据库成为产品。因此下面这些问题的实现决策就变得很重要了：(1) 网络的规模；(2) 分布式的设计；(3) 查询处理；(4) 事务处理；(5) 支持分布式数据库的操作系统；(6) 多数据库系统。这些问题的实现决策不仅影响着分布式数据库技术的发展，而且还影响着分布式数据库技术向产品的转化。

1. 网络的规模问题

目前，数据库界的人们并没完全理解所有实现分布式 DBMS 所做设计选择的含义。特别是一些协议和算法的规模会随着系统地理上的分布[2]和系统成份的增加[3]而增加。

一个值得注意的方面是：分布式事务处理机制在分布式数据库系统中是建立在广域网基础之上的。这就涉及到两阶段上锁和两阶段提交协议的问题。更重要的是与这些协议有关的一些问题，而且在一个较慢的广域网上实现起来可能比较困难。

规模问题仅是一个更为一般的问题的一个部分：没有很好地处理网络的体系结构和协议在分布式 DBMS 执行中所扮演的角色。在几乎所有的执行性能研究中，都是以一个假设的、非常简单的网络代价模型为基础。有时简单地使用一个固定延迟而根本不考虑网络特征，如负载、信息量、网络大小等。例如，考虑一个运行在 Ethernet 类型局域网上的分布式 DBMS，信息延迟会随着网络负载的增加而增加，而且并没有施加限制。因此，在 Ethernet 网上实现的分布式 DBMS 的执行性能模型就不能使用一个不变的网络延迟或延迟函数而不考虑网络的负载。一般说来，目前的情况是：还没有很好地理解不同局域网络体系结构中所提供的算法和协议的执行情况，就随意将它们从局域网移植到了广域网中。

处理规模问题的一个比较合适的方法是去开发出一个一般的、强有力的执行模型和测量工具以及相关的方法学。尽管已为集中式的 DBMS 作了很多这样的工作，但还没有扩充到分布式的 DBMS 上。

现在的分布式 DBMS 的执行性能研究通常是以简单的模型、人工的网络负载等假设为基础，或者是只考虑了仅有的几种特殊的算法。为了使研究具有一般化就需要作进一步的努力，但这并不是说不要分清主次矛盾。目前，一些系统的权衡已经处理得比较好了。例如，SDD-1 系统的查询处理就能在较慢的广域网上有效地执行分布式操作。最近的一些研究正在考虑查询处理器在较快的局域网上进行优化的问题。但是这样的一些权衡不能仅考虑定性的方面，还需要对执行模型进行定量的研究。

2. 分布式的设计

分布式数据库系统的设计方法因系统体系结构

的强有力支持；

• 基于代理者[18]的对象标识；

• 支持值：MIDS 对值概念有自己独特的理解和支持；

• 强类型的；

• Object，Frame 和 Rule 相统一的对象数据模型；

• 智能的：支持知识的表示和处理，以及基于规则的推理；

• 主动的：Rule 与 Trigger 统一，支持触发机制。

（参考文献共 18 篇略）

不同而不同。目前，有各种体系结构可供选择，如图1所示。对紧密集成的分布式数据库来说，是一个自上而下的、从需求分析和全局数据库的逻辑设计到局域数据库的物理设计的过程。对分布式的多数据库系统来说，是一个自下而上且涉及到现存数据库集成问题的过程。在这里，我们集中考虑自上而下的设计过程。

在自上而下的过程中，最使我们感兴趣的一步是分布式设计，涉及到将全局实体分布到各节点上以便设计局部概念模式。全局实体是在全局概念模式内确定的。在关系模型中，全局和局部实体都是关系，这样分布式设计就是将全局关系映射到局部关系之上。在这里，值得进一步研究的是：分布式设计的方法以及怎样将该方法集成到一个一般的数据模型化过程之中。

分布式设计的两个重要方面是分段和定位。分段是将每个全局关系分成若干段关系的集合，定位则是将这些局部关系分布到分布式系统的节点上。分段的研究主要集中于全局关系的横向（或选择）分段和纵向（或投影）分段。现在已经提出了许多定位算法。

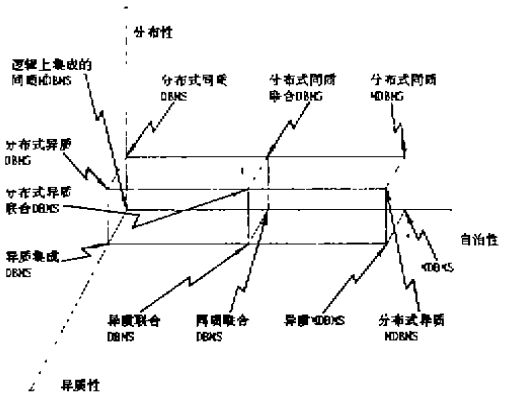

图1　DBMS实现的选择方向

目前，虽然已完全独立开发出了横向的和纵向的分割技术，但还没有基本的设计方法来将两者组合起来。一些研究人员指出，最符合实际的分段是混合分段，即横向和纵向的关系分割，但目前还未取得重大进展。在分布式设计方法中需要一个包括横向和纵向分段的算法，并将其作为一个更为一般的策略中的一部分。这样的方法将用一个全局关系和设计标准集合来产生分段集合，使得其中一部分来自横向分割，而另一部分来自纵向分割。

定位是分布式设计的第二方面，可以与分段分开来单独对待。因此，这样的过程就是线性的了，即分段的输出是定位的输入。这种处理的方法似乎减小了决策空间而简化了问题，但实际上加重了定位模型的复杂度。其实，这两个步骤具有同样的输入，区别仅是分段集中处理全局关系而定位则着重考虑段（局部）关系。这两个步骤都需要与用户应用相关的信息，如用户访问数据的频率以及数据对象的内部联系。分段和定位应重视怎样使用这些输入。分段算法决定怎样去分割关系，部分地依赖于应用的访问；定位模型则重视这部分输入在分段中所扮演的角色。因此，定位模型必须详细说明所有段关系以及用户怎样访问它们之间的联系。一个简化一些的问题是去研究分段和定位决策的内部依赖性，这就需要扩充现在的分布式设计策略[4]。

尽管集成的方法可能非常复杂，但是将这两个步骤结合在一起却可能开发出一个启发式的解决方法。一些研究表明这样的集成方法和合理的解法机制是完全可以开发出来的。在这些研究中，研究人员建立了一个分布式DBMS的模拟模型，并用特定的数据库设计来作为输入以测量其有效性。用这样的方式来开发辅助工具可能是解决设计问题的最好方法。

3. 查询处理

分布式查询处理器将一个分布式数据库（用户认为它是单个的数据库）上的高级查询翻译成局部数据库上的一个有效的低级执行计划。这样的翻译有两个重要的方面。第一，翻译必须产生输入查询的正确表示以便执行计划能产生预期的结果。这方面的形式化基础是关系演算、关系代数、以及和关系代数相关的转换规则之间的等价性。第二，必须对执行计划进行优化；即必须对代价函数进行最小化，因而查询处理器必须调查等价的其它执行计划，以便选择一个最好的。

由于一起说明这两个方面比较困难，可将它们孤立成两个顺序的步骤：数据的局部化和全局优化[5]。这两个步骤一般先经过查询分解，简化输入查询；然后用关系代数进行重写。数据局部化是将一个分布式数据库上的代数查询转换成一个等价的段查询（一个表达在数据库段上的、不同节点上的查询），并可以通过代数转换来作进一步的简化。

全局查询优化通过决策操作的顺序、节点间的数据移动，以及数据库操作的分布和局部算法的选择来为输入的分段查询产生一个优化的执行计划。

全局优化会产生许多问题，如代价模型的限制、优化代价与执行代价的权衡、优化和重优化的间隔等。

代价模型集中于全局查询优化，在考虑访问的方法时提供必要的分布式 DBMS 执行系统的抽象模型，在考虑物理模型信息和有关的统计时提供一个数据库的抽象模型。代价模型是用来预测一个查询的执行计划的执行代价，一些重要的限制与代价模型有关，这就限制了在提供完整的输出中进行优化的效果。在参数化代价模型时扩展查询优化的工作是很有用的，而且代价模型也可以通过实验来做不断的改进。

尽管查询语言的能力在不断地加强（如新版本的 SQL），但是全局查询优化主要集中在查询语言的子集上，即带有连接谓词的 Select-Project-Join 的查询。这样重要的一类查询可以进行有效地优化。其它的一个重要查询如带有 Unions、Fixpoints、Sorting 的查询也需要优化。一个较好的解决办法是将语言的理解和它本身的优化分开来，以便使优化由几个专门的优化模块来加以完成。

优化代价与执行计划代价的权衡是非常必要的。较高的优化代价对于为重复使用查询而产生较好的执行计划是可以接受的，这些执行计划将减少查询的执行代价并通过多次执行而分期偿还了优化的代价。但是较高的代价对仅执行一次的查询是不能接受的。优化的代价主要花费在为各种执行计划寻找解空间。在一个分布式系统中，解空间可能非常大，有各种各样的分布式执行策略，因此，开发出有效的执行策略而避免穷尽式的搜索方式是非常重要的。

全局优化主要是在查询执行前进行的，也称作静态优化。这种方法的一个主要问题是：由于分段大小的变化或数据库的重组（这对负载平衡非常重要）可能使得优化代价模型变得不够准确。因而，很关键的问题就是决策查询的重新编译以及重新优化的间隔，权衡优化代价与执行代价。

4. 事务处理

分布式的事务处理尽管已经过了广泛的研究，但仍存在一些重要的问题。前面已经讨论了规模问题，下面主要考虑拷贝控制协议、更复杂的事务模型、以及非序列化的正确标准。在冗余的分布式 DBMS 中数据库的操作是在特定逻辑数据对象上进行的。这里使用术语“数据对象”来代替普通的“数据项”，是因为不想去考虑数据的粒度。拷贝控制协议是负责将一个逻辑数据对象上的操作映射到这个数据对象的多个物理备份上的操作，能保证重复数据库的相互一致性。此外，使用 ROWA 规则（读一个，写全部）是强行使之具有一致性的最直接手段。因此，如果每个数据对象的所有备份都有统一的值，那么该重复数据库就处于相互一致的状态了。

数据域的重复需要进一步的研究，尤其是有关计算和通讯的重复方法以及与特定应用有关的系统利用等方面的问题。还需要评价各种算法和系统设计人员所作的承诺，并用一个一致的结构来比较各种技术。定量地评价重复技术的一个困难就是缺少可普遍接受的失败事故模型。例如，Markov 模型，总是假设单个失败事故具有统计独立性而且几乎与节点失败的网络分割无关，然后在这个假设的基础上通过重复来进行分析。目前，还不知道这些假设能否站得住脚，也不知道 Markov 模型对这些假设有多么敏感。在缺少经验度量的条件下，模拟 Markov 模型的有效性是不可靠的，因为这样的模拟通常具体化了奠定 Markov 分析的那些假设。因而，我们需要经验来研究在现实生产系统中监控失败的模型，并且需要构造一个简单而又典型的失败负载模型。

为了达到数据重复的双重目标——可用性和执行性能——必须提供一个集成的系统并且使用计算和通讯（包括 I/O）的重复来共同处理数据的重复。但现在仅集中研究了数据重复；几乎还没有对计算和通讯的重复进行什么研究。目前计算重复的研究主要是设置中的一些方面，包括作为热点备用（hot standbys）的同步双工处理的执行，以及防止人为设计错误的相同软件的不同版本实现等。通讯信息的重复，基本上是在提供可靠的信息传送方面进行研究，目前已经有几篇论文报告了有关使用 I/O 信息的重复来增加事务系统的可用性的情况。但是，还需很多工作来进一步研究怎样将这些工具和数据重复集成起来以便支持实时控制系统这样的应用。这方面的工作不仅有助于保护操作系统而且有助于开发合适的工具集来支持容错系统。

另外，对于重复及其相关的一些方面来说，我们需要更多精心设计的事务模型，尤其是需要那些利用应用语义来达到具有较高可用性、执行性能和并发性的模型。随着数据库技术不断地进入新的领域如工程设计、软件开发、以及办公信息系统等，事务的本质和需求也在不断地变化，因而就需要将这方面的工作建立在正确性而非连续性的标准之上。

作为一个方向，事务模型可以沿着两个轴向来分类：事务的结构和事务操作对象的结构。沿着事务

的结构这条轴线。我们认识到了平滑（flat）事务、封闭式嵌套（closed-nested）事务、开放式嵌套（open-nested）事务，以及包括开放和封闭的嵌套。这样，我们增加了序（order）这个复杂的概念，从而增加了模型的复杂度。沿着对象的结构这条轴线，我们确定了简单的对象（如页）以及作为抽象数据类型实例的对象。这样再一次增加了模型的复杂度。之所以区别后两种对象是为了表明作为抽象数据类型的实例的对象支持封装。因而比简单的对象更难进行事务处理，但它并无复杂的结构。不包含其它对象，而且其类型也并不涉及固有的层次结构。

在这个框架中，大多数分布式系统中的事务模型集中于描述简单对象上执行的平滑（flat）事务。这一点在设计中是很容易理解的。目前在简单对象上的嵌套式（nested）事务模型中已做了一些工作，但还有许多工作有待完成。特别是在分布式系统中，这些事务模型的语义仍在研究之中。另外，将简单的处理应用到作为抽象数据类型实例的对象和复杂对象的有关研究已经完成了。应反复进行这些试验性的研究以确定其全部语义，并将之结合到一个 DBMS 和恢复管理者进行交互和分布特性之中。

复杂的事务模型之所以重要，主要有两个原因：第一，这些模型的语义可能有助于分布式多数据库系统中事务处理的研究。第二，为了使分布式 DBMS 支持一些新的应用领域（如工程设计，办公信息系统、计算机辅助设计等），需要将复杂数据上更为抽象的操作结合到事务模型之中。这些新的应用有一些共同的特点与我们通常熟悉的数据库访问有所不同。例如。计算机辅助设计的工作环境需要参与者在访问共享资源时去合作而不是竞争，这样的变化就迫使我们去开发新的事务模型和正确的标准。

5. 操作系统

就当前的情况看，我们并不想将集中式或分布式 DBMS 作为一个普通的应用来运行在主操作系统的上面一层[6][7]。DBMS 的需求与当前操作系统的功能不相匹配。这种不一致性在分布式的情况下更为突出。现在的分布式操作系统并不提供它所需要的功能。如，分布式事务支持的并发控制和恢复，分布式永久数据的有效管理，以及更复杂的访问方法。另外，分布式 DBMS 还要求分布式操作系统在怎样执行其传统功能方面作一些修改（例如，任务调度、命名、以及缓冲区的管理等）[5]。在这一部分，我们将着重讨论 DBMS 分布式操作系统集成中的基本问题，系统体系结构、资源的透明命名、永久数据的管理、以及对事务的支持。

一个重要的体系结构方面的考虑是分布式 DBMS 与分布式操作系统的耦合而不是一个二元集成、以及通讯网络协议所增加的问题的复杂性。因此，这就要求体系结构必须灵活地适应分布式 DBMS 的功能、分布式操作系统的服务、以及通讯协议标准模型（如 ISO/OSI、IEEE802）。在这种情况下，操作系统的核中包括太多的数据库功能或修改太多的操作系统功能可能并不是什么好的解决办法。一个可行的办法是：操作系统仅实现那些最本质的操作系统服务以及那些能有效实现的 DBMS 功能。最能满足这种需求的模型可能就是客户/服务器这种体系结构了，这种结构由一个核来有效地提供数据库的功能而且不妨碍 DBMS 在用户级别上有效地实现其它服务（如 Mach 和 Amoeba 操作系统）。面向对象的技术也有助于这样的系统结构的集成。

命名是为操作系统提供对系统资源进行透明性访问的基本机制。是否应该在操作系统的级别上就使分布式对象的访问是透明的？这是一个有争议的问题。它涉及到数据管理的灵活性、易使用性、以及系统管理费用之间的权衡。对一个分布式 DBMS 来说，透明性是很重要的。许多分布式 DBMS 企图建立起自己的透明命名计划而没什么进展。命名问题、分布式的访问、以及操作系统命名服务器之间的关系需要进一步的研究。一种较为有价值的命名机制是较早的系统（如 Hydra）中使用过的且现在的操作系统（如 Amoeba）还在使用的机制。

永久数据的存贮和管理是 DBMS 的基本功能。这里所说的永久数据是指那些寿命比操作它的程序执行时间要长的数据。操作系统的传统处理方法是使用文件系统来处理永久数据。目前的情况有可能设计出一个集成系统，使得 DBMS 作为操作系统的文件系统。在更为一般的层次上，就需要进一步研究程序设计语言的合作、操作系统、以及管理永久数据的 DBMS。分布式文件系统并不能解决分布式 DBMS 的问题，因为它们既不能提供对数据进行并发访问的控制也不能提供共享（共享的粒度太大了）。

在分布式操作系统[illegible]广泛研究了两个通讯方面的问题：信息传送和远程过程调用（RPC）。在这里，信息传送是指逻辑的而不是物理的。RPC 必须在各节点间作为物理信息来传送。简单的 RPC 语义（即发块和一次性执行）很适合于分布式系统的设计。因此，有些研究人员建议[8]，在用户级别上用 RPC 对分布式数据的访问来全面代替透明性访问。但是在一

个异质环境中实现 RPC 机制却并不容易．主要的问题是不同厂家的 RPC 系统并不是可互操作的．因而有必要在较高的层次上对通讯进行抽象以克服异质的问题．或者在较低的层次上如信息传送等方面进行抽象以达到更大程度的并行。这两种之间的权衡还需进一步的研究。

在当前的 DBMS 中．事务管理者是作为 DBMS 的一部分来实现的。能否将事务作为标准操作系统的一个服务来实现已经作了许多讨论．有主张的、也有反对的。但这个问题的明确解决还需要更多的研究和各种目的（非 DBMS）的事务管理服务的经验。

6. 多数据库系统

多数据库系统的组织是在逻辑上集成分布式数据库的一种方法．与单数据库系统的主要差别是每个节点上的数据管理者所担负的自治程度。单数据库系统的各组成部分是一起合作来进行工作的．而多数据库系统的各组成部分则没有这种合作的概念。特别是这些组成部分可能是独立于 DBMS 的．例如它们本身就可能有执行事务的功能．但它们不能执行跨越多个组成部分的分布式事务。在这部分我们将主要讨论多数据库系统中和查询处理以及事务管理有关的开放性问题．也将考虑标准化和它们在互操作中所扮演的角色。

潜在的异质和组成部分的自治产生了查询处理以及查询优化的问题．主要的问题是：由于不知道局部代价函数或者局部代价值不能传送给多数据库管理系统（MDBMS），使得全局优化变得非常困难。有些人认为[9]基于定性信息的语义优化可能是能得到的最好结果．但目前并没有全部理解语义查询处理的细节。一个可能的、尽管不是最好的方法是开发出一个层次结构化的查询优化以便先进行一些全局查询优化．然后让每个局部系统在局部化的子查询上做进一步的优化．下面所要讨论的标准可能更容易共享一些代价信息。

基本的 DBMS 的自治性使得多数据库系统中的事务处理变得非常困难。由于自治性．它们就有自己的事务处理服务（事务管理者、调度者、恢复管理者等）而且能接受局部事务并进行处理。MDBMS 层有自己的事务处理组成部分来负责接受和协调访问多数据库的全局事务。一个全局事务分成若干子事务．并将每个子事务提供给组成 DBMS 的相应部分。但是．由于 MDBMS 并没有意识到局部事务．因而不能控制局部冲突．也不能控制由于局部事务的冲突而引起的全局变易中的间接冲突。

目前已经提出了一些方法来处理并发的多数据库事务处理。有些使用事务的全局序列化作为其正确性标准．而另一些则放宽了对这种序列化的需求。目前．这方面的工作集中在该问题的理解和形式化．但这只能作为初步尝试．还有许多问题需要研究。研究的一个领域就是事务模型中的修正以及正确性标准。尽管现在已经作了一些努力来重新建立事务模型的假设．但还需很多工作。嵌套式（nested）事务模型总的说来对多数据库系统很适合．而且它的语义也是基于有关事务行为知识的．当然该语义还需进一步形式化。因而．我们要重新考虑多数据库系统中一致性的含义．一个较好的出发点是由 Gray[1]定义的一致性的四个级别。

另一个要进一步研究的难题是多数据库系统的可靠性和恢复性．单个 DBMS 的自治性使得将两阶段提交结合到全局事务处理中变得非常困难．因而难于使分布式事务成为事务的基本单元．尽管不少研究人员最近对此作了努力，但他们的方法基本上是原始的工程解法．因此．多数据库系统的可靠性和恢复性协议就必须进一步开发．而且要将它们和并发控制机制集成在一起。

自治性是给多数据库系统增加复杂性的一个主要因素．进一步开发多数据库系统的主要障碍就是我们目前还没有理解自治性的本质．自治性可能包含多个因素．因而必须精确地描述出自治性的本质特征。

但令人遗憾的是．大多数人认为自治性是一个可有可无的特征．一般情况下．对自治性的分类是粗线条的．分为无自治性、半自治性、以及全自治性三类。但实际上在无自治性和全自治性之间可能包含具有各种自治程度的类型。因此．对自治性的进一步研究就很重要了．这可以分为下面几步：首先．定义什么是“无自治性”以及“全自治性”；其次．描述和定义许多不同的自治级别；第三．为每个自治性级别确定数据库的一致性程度。在这一点上就需要在不同级别上讨论不同的事务模型和执行语义．从而为自治性的互操作性和可能的异质数据库确定结构化的层次．这类似于 ISO/OSI 模型．这样的一种结构将允许我们在不同的级别上确定接口标准．目前．已经在远程数据访问（RDA）标准下做了一些工作．这会给互操作性问题提供一个切合实际的解决办法。

7. 技术的进步和新的问题

下一代数据库系统的一个共同特征是它们将需要一个比关系模型更强有力的数据模型，而且还不

失去其优点（即数据独立性和高级查询语言）。在复杂的应用中，如CAD/CAM、软件设计、办公信息系统、以及专家系统等，关系模型显示出在复杂对象支持、类型系统、以及规则管理等方面的局限性。为了说明这些问题，现在一些人员正在研究两个很重要的技术：知识库和面向对象的数据库。随着功能的不断增加，另一个主要的问题将是系统的执行性能。在多处理器计算机中使用并行算法将是提高执行性能的一种很有希望的办法。因此，要进一步开发新的分布式数据库技术以便应用到实现并行的数据库系统之中。

面向对象的DBMS将数据库技术和面向对象的程序设计技术结合起来。这种方法给大数据量的应用提供了更强的模型化能力和灵活性。

在过去一段时间里，OODBMS已经成为集中研究和实现的目标，并产生了大量原型和商品化系统。尽管如此，开发分布式的OODBMS的理论和实践却还没有发展成熟。这主要是分布式环境使问题变得更为复杂和困难。而且数据字典的管理以及分布式对象的管理等问题还没有得到解决。但是，分布式是一个必须解决的问题，因为网络化的工作站环境中需要OODBMS技术的支持。在较早的商品化OODBMS中（如，Servio Corp. 的Gemstone），使用了客户/服务器（Client/Server）体系结构，其中多个工作站可以访问服务器上的集中式（集成化）数据库。但是，在一个工作站和服务器的网络内部分布一个面向对象的数据库则更具有吸引力。实际上，目前已有OODBMS支持某种形式的数据分布的透明性了（如，Ontologic Inc. Ontos和MCC的原型的Distributed Orion）。

知识库管理系统通过管理知识和数据的使用来使数据库的管理更加智能化。目前，已经对演绎数据库作了大量研究。这种数据库是以规则的形式来获取知识的特定知识库系统。演绎数据库系统管理和处理着数据库的大量数据所确定的规则，而不是在一个独立的子系统中进行这样的管理和处理。规则可以是说明性的（断言），也可以是命令性的（触发器）。在这里，规则的管理是本质性的，因为它提供了一种统一的方式来处理语义、视口、保护、演绎、以及触发器的一致性控制。在演绎数据库中，现在的主要工作集中于规则程序的语义以及演绎查询的处理，尤其是递归和否定谓词。但是，还需要做许多工作将规则支持和面向对象的能力结合起来。

由于和OODBMS应用具有同样的原因，因而也需要将知识库的技术应用到网络化的工作站环境之中。在数据库由一个多处理器的数据库服务器来管理时，也可以将知识库的应用系统放在并行计算环境中加以处理。一般情况下，可以用分布式关系数据库技术来简化许多问题。不象大多数的OODBMS方法（它企图开发出一种面向对象的编程语言），这种相似性将给在分布式环境中实现知识库带来较大的优越性。因此，所面临的新问题主要是分布式知识管理和分布式知识库的查询处理，而不是分布式数据的管理。

并行数据库的设计主要是试图利用最近发展起来的多处理器计算机体系结构来建立高效率能容错的数据库服务器[10]。例如，通过在多个节点上对数据库进行分段，就可以获得更大程度上的子查询并行性。现在，这个领域的大多数工作集中于支持SQL，因为这方面的努力主要试图解决面向集合的处理和可移植性的应用问题。正在研究的一个问题是：是共享式存贮器还是分布式存贮器的多处理器体系结构更适合于数据管理？该问题的解决还需要进行更多的实验和努力。

并行数据库系统的设计问题，如操作系统的支持、数据的分布、并行算法、以及并行编译等，对这两种体系结构来说都是一样的。一些研究人员认为并行数据服务器的较为可行的环境是将多个处理器组放置在主干网上，并且由这些处理器构成分布式系统。这样一种环境中的问题是内联网络。特别是，当执行跨越多个（甚至可能是异质的）处理器组的数据库命名时，就至少会产生在分布式多数据库系统下曾遇到的问题。而且，查询的优化不仅是为了在服务器组上的并行执行而且也是为了跨越网络的执行。

分布式数据库系统的原定目标：更容易更经济的系统扩充、分布在各节点而又重复的数据的透明性、由分布式事务的方法所改进的系统可靠性、由子查询所增进的系统执行效率，目前已经在一些商品化的系统中不同程度地实现了，但是要全面实现这些目标却还有许多问题必须解决。下面是一些研究方向：

（1）需要研究执行模型和方法对建立在基本技术上的算法和机制的敏感性。

（2）需要研究怎样利用分布式查询处理机制去进行比Select-Project-Join更复杂的查询，而且还需用代价模型去决定何时进行建立在预处理基础之上的多个查询处理是最有效的。

计算机科学 1994Vol. 21№. 2

下一世纪软件工程展望

陈沐天 许 创 编译

摘要 本文主要讨论软件工程的未来以及基于知识的软件工程（KBSE）的进展将如何影响系统开发环境。文中所提出的新鲜而富有创见性的观点，为我们展示出一幅未来软件工程的全新面貌。

一、引言

在2000年，基于知识的软件工程（KBSE）将有巨大的潜在市场和丰富的环境。今后十年中硬件的改进将刺激大而复杂的应用软件需求，软件接口和操作系统的标准化将加强软件开发者之间的竞争。在这种形势下能够快速创建正确无误软件的开发者将兴旺发达，而那些提供有误软件、成年地延期交付使用的开发者将断绝生路。为了对付这种挑战，开发者将探求使软件设计自动化的工具及方法。

计算机辅助软件工程（CASE）的市场已有巨大增长，然而当前一代CASE工具却局限于浅层表示及浅层推理方法。CASE工具进化到或者替代以具有更深层表示、更成熟推理方法的工具，其可用技术来自AI、形式方法、程序设计语言的理论以及计算机科学的其他领域。这一技术会使目前软件开发过程中丢失的大量知识以机器编码形式进行自动掌握。KBSE将使软件设计革新，正如计算机辅助设计已使硬件设计发生革命性变化、台式出版系统使版面设计发生革命性变化一样。

二、前景

目前KBSE正处于像早期编译程序和专家系统那样的发展阶段。市场上有几十个工艺式系统作为工业试验项目在实际使用。在研究所，许多原型KBSE系统已被开发，促进了软件设计知识自动化及形式化的科学。近二十年来，程序综合已成熟到只要在人的有限指导下就可合成复杂算法的程序。在需求及规格说明工程方面、智能助手的研究还差些，但已显示出可观的前景。

2.1 今后十年

在这十年中，KBSE技术在商业上主要用在软件维护及专用程序综合方面，其影响是渐进的并且与当前软件开发方法相容。在需求及规格说明工程方面的智能助手型的研究系统（如ARIE和ROSE-2）加进许多当代的CASE表示。随着它的成熟，会被集成为下一代CASE工具。在这十年，突破点将是通用程序综合系统，对它只要给出专业理论后它会以交互方式开发一个程序且比手工更快。该突破的关键将是继续改进搜索控制。重大研究计划正在进行，KBSE技术将从小规模程序设计扩大到大规模程序设计。

2.2 下一世纪

软件工程将演化成面目一新的学科，软件将变

（3）需要进一步研究高级的事务模型，以便使之能更好地反映出处理模式的共性。

（4）需要进一步分析重复对分布式数据库系统体系结构和算法的影响。需进一步开发出能改进系统有效性的拷贝控制协议。

（5）需要开发出一个更好的接口以及有分布式操作系统合作的分布式DBMS实现策略。

（6）需要在理论上进一步探究和修正分布式数据库的设计方法。

（7）需要解决与自治性有关的问题。

分布式DBMS技术的进步使得并行数据库服务器更加灵活，而且这将从两个方面来影响分布式的数据库系统，首先，在这些并行数据库服务器上实现分布式DBMS将需要现存的并行机上的算法和协议。其次，并行数据库服务器将作为网络服务器和网络相联，这就需要进一步开发处理数据管理层次结构的分布式DBMS。随着分布式数据库技术对非商业数据处理领域的渗入，这些系统的需求能力也随着变化，强调关系的系统转向更强有力的数据模型。当前主要的研究是集中在面向对象的数据库系统和知识库系统。

已经看到了，许多重要的技术问题等待解决，而且新的问题还会随着分布式数据库的发展而产生，这些问题还需要我们做大量的努力。

（参考文献共11篇略）