工程数据库管理系统的体系结构*

陈 俊 孙建伶 董金祥

(浙江大学计算机科学与工程系 杭州 310027)
(浙江大学CAD & CG国家重点实验室 杭州 310027)

摘要 设计工程数据库管理系统的体系结构需要在两个重大问题上作出选择:①采用何种客户/服务器结构,以便在客户机和服务器之间寻求一个合适的功能分配点;②采用何种并发控制策略来保证客户端缓冲数据的一致性.首先讨论3种不同的客户/服务器结构(即对象服务器、页面服务器和文件服务器),比较它们之间的差别和各自的优缺点;其次讨论了实现缓冲一致性的各种并发控制策略;最后介绍了工程数据库管理系统OSCAR的体系结构.

关键词 工程数据库管理系统,体系结构,缓冲一致性,并发控制.

中图法分类号 TP391

工程数据库的应用环境具有与商业数据库的应用环境所不同的特点.商业应用的特点是事务短、事务吞吐量大、并发度要求高.工程应用的特点是事务长、客户处理的数据局部性好、大部分应用在客户端完成.工程数据库为了适应工程应用的特点,在体系结构上就不同于商业数据库.

工程数据库一般采用客户/服务器(Client/Server)模式,但和商业数据库的客户/服务器模式是有区别的.

客户/服务器DBMS可分为两大类:传送查询(Query-shipping)与传送数据(Data-shipping)[1].两者的区别在于传送查询系统中客户向服务器发的是查询请求,客户与服务器之间传送的是查询请求和查询结果;传送数据系统中客户向服务器请求的是具体的数据,而不是查询结果.在传统的关系数据库中,传送查询的体系结构很好地满足了商务方面的应用要求.作为适应大型工程应用环境而出现的面向对象数据库管理系统(OODBMS)[2],在很大程度上不同于关系数据库.几乎所有产品化的OODBMS(包括O2[3],ObjectStore[4],ORION[5])采用了传送数据的体系结构.传送数据的体系结构又可分为3类[6]:对象服务器(Object-server)、页面服务器(Page-server)和文件服务器(File-server).

随着高性能工作站的日益普及和网络传输速率的提高,计算工作量从服务器到客户机的下载已成为未来计算环境发展的趋势,传送数据的3种体系结构实际就是在考虑了应用要求和硬件发展的基础上,为了在客户机和服务器之间寻求一个合适的功能分配点而形成的3种方案.

本文第1节分别介绍客户/服务器的3种结构,讨论它们的结构、功能、特点、性能比较.第2节讨论在客户/服务器DBMS中存在的缓冲一致性(Cache Consistency)问题及相应的并发控制方法.第3节介绍我们自主开发的工程数据库管理系统OSCAR在体系结构上的特点.最后是总结.

1 工程数据库的3种客户/服务器结构

1.1 对象服务器(Object-server)

(1) 对象服务器的结构与基本功能 对象服务器的结构如图1所示.工作站与服务器之间传送的是对象,

* 本文研究得到国家自然科学基金和浙江省自然科学基金资助.
作者陈俊,1974年生,硕士生,主要研究领域为工程数据库.孙建伶,1964年生,博士,副教授,主要研究领域为数据库.董金祥,1945年生,教授,博士生导师,主要研究领域为工程数据库,CAD.

本文通讯联系人:孙建伶,杭州 310027,浙江大学计算机科学与工程系

本文1997-12-23收到原稿,1998-02-25收到修改稿

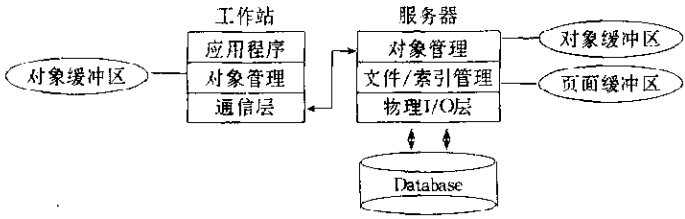

图1

工作站实际把服务器看成一个对象存储器,它不知道任何有关对象的物理存储的细节.工作站与服务器重复了对象管理功能.

工作站端的对象缓冲区并不是必须的,但是为了性能上的考虑,在客户端设置缓冲区是可取的.在客户端和服务器同时存在缓冲区会带来缓冲一致性问题.后面将看到,同样的问题在页面服务器中也有.有关缓冲一致性的问题将在第2节集中讨论.

当客户端需要一个对象时,它先查看本地的对象缓冲区,若没找到,就向服务器发一个需要该对象的请求.服务器收到请求后,或直接将其缓冲区中的对象返回给客户,或在尚未缓冲该对象的情况下先从磁盘缓冲该对象(服务器也可能将磁盘上的对象不经缓冲直接读出返回给客户,这取决于服务器采取的缓冲策略)[7].

(2) *对象服务器的特点* 对象服务器的核心特点是服务器端也有对象的模式信息,并且它提供给客户的也纯粹是对象服务.对象服务器具有的所有实现上和性能上的细节特征全部基于这个特点.

服务器端保留有对象信息意味着方法可以在服务器端运行,在一个大集合中查询一小部分对象时,该特点意味着查询方法可先在服务器端运行,这样就大大降低了网络传输的负担.

对象服务器的并发控制实现很方便.因为服务器有对象的模式信息,所以,并发控制可以是完全集中式的,而且对象级锁的实现也很自然,这在并发度高的应用环境中是一个相当有竞争力的特点.日志的实现也很灵活.日志的类型可以是物理的,也可以是逻辑的(在后面可以看到,页面服务器和文件服务器只能采用物理日志).这种体系结构下的恢复方案实现可以有多种选择:工作站可以在自己的缓冲区中操纵数据和生成日志,然后把两者都传送给服务器;也可以只生成日志,修改的工作由服务器来做.

客户向服务器请求的数据单位是对象.这虽然给客户端的应用带来方便,但也有不足之处,即在最坏的情况下,客户每请求一个对象都要进行一次网络传输,这种频繁的昂贵操作会迅速成为整个系统的瓶颈.

对象服务器的工程数据库系统在服务器端的模块相对庞大和复杂,在工作站功能日益强大的今天,这是一个不符合体系结构发展趋势的特点.

(3) *对象服务器的性能分析* 总体上看,对象服务器在局部性好(或较好)的操作时性能很好,随机访问(特别是随机更新)性能不错,但局部性差的操作比页面服务器和文件服务器慢很多,特别在顺序访问大量小对象时更是如此.

对象服务器受聚簇的影响非常微小,在各种类型的访问中,聚簇率(Clustering Factor:是一个百分数.若聚簇率是f,则对象A引用的所有对象中,有$f\%$的对象聚簇在对象A的附近)的大幅度提高几乎不会带来任何性能上的改进.原因是当聚簇针对服务器的磁盘尽可能紧凑地存储相关对象的同时,工作站与服务器之间的数据传输机制却没有利用这个有利的特性.实际上,当对象在服务器端从Page Buffer转到Object Buffer时,就已失去了这一聚簇信息.工作站的缓冲区大小对对象服务器的顺序操作没有影响,而对随机读操作性能的提高会有一个阈值.缓冲区大小在该阈值以下时,缓冲区的增大带来较明显的性能改进,但大于该阈值后性能不再提高.原因是低于阈值时影响性能的瓶颈是缓冲区的替换动作,高于阈值时,瓶颈成为每次请求对象时的RPC调用.在随机修改时,当聚簇率较低、工作站缓冲区较小时,对象服务器比页面服务器和文件服务器要好得多,这是因为它没有写任何"多余"的数据.和随机读相同,在客户端的缓冲区大到一定程度后性能不再改进,所以,随着聚簇率的提高和缓冲区的增大,后两种体系结构(特别是页面服务器)的性能超过了对象服务器.

把所有的操作综合起来,对象服务器确实随工作站缓冲区的增大而性能有平缓的改进,但受其影响程度不如后两者.在聚簇率和客户端缓冲区这两个因素好到一定程度后,它的性能落在了后面.

1.2 页面服务器(Page-server)

(1) *页面服务器的结构和基本功能* 页面服务器的结构如图2所示.可以看到,在页面服务器中,服务器没有对象管理模块,这使页面服务器和对象服务器有某种"本质"上的区别,因为服务器不再知道任何有关对象的

逻辑模式信息.在这种情况下,服务器实际上充当了一个有并发控制和恢复功能的虚拟页面存储器,工作站和服务器之间传输的也不再是对象,而是页面[8].

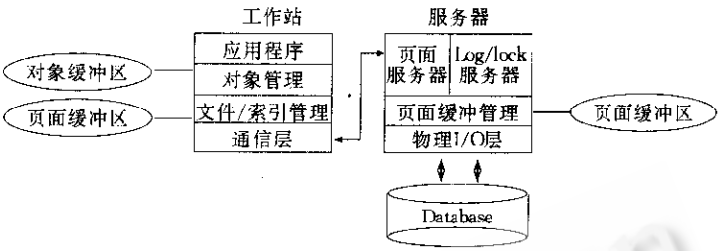

图2

页面服务器中服务器基本上由一个I/O层、一个大的页面缓冲器以及上层的并发控制和恢复模块构成.当工作站向服务器请求一页时,服务器先给页面上合适的锁,然后从缓冲区将页面传回(若页面未被缓冲,则先要从磁盘读入),工作站的对象缓冲区也可以不设,依设计上的需求而定.

(2) 页面服务器的特点　在页面服务器中,DBMS中的相当一部分工作移到了工作站上,服务器只管理页面缓冲和并发控制、恢复,这使工作站的独立性大大增强.同时,服务器负担的下降意味着一个服务器可以带许多客户节点,虽然即使只请求一个几十字节的小对象也要传送一页,但实际上传4K字节比传几十个字节多花不了多少时间,而且传过去的"多余"的数据恰恰是聚簇起巨大作用的前提.

在这些诱人的特性之外,页面服务器也为其服务器端设计上的简单性付出了代价.一个页面服务器的设计很大程度上就是如何利用其他的补偿手段来降低这类代价的过程.

因为方法不可能在服务器端运行,所以一次只查询几个小对象的操作有可能需要一大群页面,引入索引可以减少这类开销.另外,由于服务器没有对象的模式信息,所以,对象级锁很难实现.因而,大多数页面服务器对数据采用粒度为页的严格二阶段锁协议,放弃了对象级锁.在锁的粒度为页的情况下,放有全局索引的页面有可能成为系统性能的瓶颈.如何用其他手段来提高索引页的并发度是系统设计上必须考虑的一个重点.

恢复问题需重新考虑.[1]在页面服务器中,日志由客户端生成,而由客户端传回的日志页和脏数据页之间的对应关系原则上是无法精确跟踪的(这个问题在对象服务器中是不存在的),但这个问题并不如想象的那么难,使用某种算法能保证类似保守先写日志的协议.还有,由于服务器不知道对象的模式,而恢复又要在服务器端执行,所以,逻辑日志在这种体系结构中是不可能实现的.

另外,注意到模式信息在客户端解释并不意味着数据库容易被破坏,因为客户端的应用程序对数据库的操作界面是由客户端的系统模块向上提供的,并不能直接存取数据库的模式信息.

(3) 页面服务器的性能分析　在顺序访问时,若页面间的数据(对象)没有交叉引用,其性能非常好,接近单用户的DBMS;若有交叉引用,则性能有所下降.

随机读时,在聚簇率很低时,页面服务器很慢,逊于其他二者,原因是它频繁地使用代价高的RPC操作(它比文件服务器慢是因为NFS读比RPC读要快).但是,随着聚簇率的提高,页面服务器性能提高很快,迅速超过了对象服务器.这和页面服务器更多依赖于页面内数据的局部性有关.

随机更新时,页面服务器的性能虽然随聚簇率的上升而提高,但总体上仍不如对象服务器(但比文件服务器快,因为NFS写是很慢的),原因如前所述,即对象服务器不写"多余"的数据.

页面服务器对聚簇范围(Cluster Region:是一个页面数.若聚簇范围是r,且对象A引用了对象B,对象B存储在对象A周围的r个页面范围内,则称对象B聚簇在对象A的附近)和客户端的缓冲区大小非常敏感,页面服务器的结构表明其性能很大程度上依赖于数据的局部性.所以,上述两个因素在页面服务器中是起关键作用的,适度改进聚簇算法和加大客户端缓冲区可以为页面服务器带来性能上的飞跃.

1.3　文件服务器(File-server)

(1) 文件服务器的结构与基本功能　文件服务器的结构如图3所示.文件服务器实际上是页面服务器的一种变型,工作站使用某种远程文件服务(如NFS)来直接读写服务器的一页,所以,文件服务器的服务器端模块变得异常简单,甚至比页面服务器还简单得多,在NFS包办一切物理操作的前提下,文件服务器略去了缓冲管理和物理I/O层.

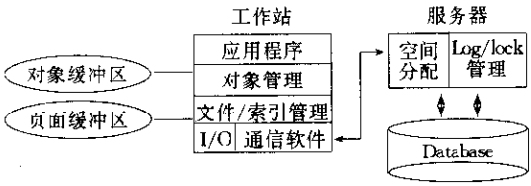

图3

(2) 文件服务器的特点　文件服务器因为其结构上与页面服务器具有相似性,所以它有页面服务器的所有特点.另外,它的独特在于使用了某种绕过DBMS的数据读/写手段.这种"绕过"的特性使文件服务器除了具有页面服务器的优缺点外又有自己的长处和短处.

由于文件服务器使用了直接对操作系统内核操作的NFS服务,它完全避免了用户态进程间的切换,这使得NFS读非常快;但是,基于UDP的NFS写是非常慢的,所以页面的写操作很容易成为系统的瓶颈.

(3) 文件服务器的性能分析　文件服务器的性能曲线在所有情况下都类似于页面服务器.在不需要写的操作中,其性能比页面服务器略好,写的操作比页面服务器差许多,所以其总体性能逊于页面服务器.

同页面服务器一样,文件服务器对聚簇率、聚簇范围、客户端缓冲区大小很敏感.在这三者条件不好的情况下,文件服务器比页面服务器慢,但随着这3个条件的改善,其性能逐渐接近页面服务器,直到几乎一样为止.

1.4 工程数据库的3种客户/服务器结构比较

页面服务器和文件服务器的主要优点是服务器端模块设计简单,大部分复杂工作放到工作站上去做,符合当前硬件的发展趋势.其主要缺点是粒度小的多级锁难以实现,关于索引页的非二阶段锁也是一个难点;对象服务器需要更复杂的服务器模块设计,虽然其数据传送效率不高,可能请求一个对象就要调用一次RPC,但它比前者优越的地方在于粒度小的多级锁实现简单,而且服务器端可以执行方法.

页面服务器比对象服务器还需要多考虑的一点是安全性问题.首先,页面服务器通过网络向客户提供的是页面服务,所以,实际上服务器对客户端的页面存取是无法进行权限控制的,但从另一个角度看,客户端和服务器端的模块共同组成了一个系统,客户端通过网络的页面存取是系统内部的操作,数据库的正常操作应该建立在客户端模块正确操作的基础上;其次,对于外来的网络上的破坏(或窃取)操作,系统本身没有办法防止,因此,页面服务器的环境中对网络通信模块的安全性要求就提高了.

总之,3种体系结构中没有绝对的优胜者,页面服务器在客户端缓冲区足够大,聚簇效率高的时候是一个很好的选择;对象服务器在局部性差的操作时很慢,但其性能在客户端缓冲区不大的情况下比页面服务器要好;文件服务器若纯粹依赖于NFS进行读写操作,NFS写会成为整个系统的瓶颈.

2 客户/服务器环境下的缓冲一致性问题及并发控制方法

客户/服务器的DBMS为了提高数据存取效率,一般在客户端设置一定大小的缓冲区.在页面服务器和对象服务器环境中,同时在客户端与服务器端缓冲数据会带来数据一致性问题.这样,就需要采用某种并发控制方法来保证缓冲数据的一致性.

2.1 保证缓冲一致性的并发控制方法

Basic 2PL[9](以下简称B2PL)作为一种经典的并发控制方法在传统的中心数据库中广为采用.在新的传送数据的客户/服务器环境中,这种算法经过改进,利用了事务间的局部性(Inter-transaction Locality,即相邻事务访问的对象往往有很大一部分是相同的)之后,称为Cached 2PL[10](以下简称C2PL).C2PL的一种扩充是Callback 2PL(以下简称CB2PL),它实际是在C2PL的基础上加入了Callback Locking[11]的思想.另外,2PL的变型还有乐观的二阶段锁Optimistic 2PL[16](以下简称O2PL).它们的特点如下.

B2PL:传统的2PL,事务分为上锁和释放锁两个阶段.上锁阶段不能释放锁,释放锁阶段不能上锁[9].在客户/服务器环境中的B2PL实现中,在上锁阶段,客户端向服务器发上锁请求并下载页面,在释放锁阶段,客户端回送页面并释放锁.

C2PL:同B2PL所不同的是,客户端的事务结束后,其页面仍留在客户端的缓冲区中,后来的事务可以存取这些缓冲的页面,但是要先向服务器发上锁请求.服务器判断客户端的页面是否"过时",若"过时",则需要向客户传送更新的页面,否则,传输页面的工作可以省去.

CB2PL:客户端的事务结束后不但保留其页面,而且保留页面的锁(为了避免过多的锁冲突,一般只保留读

锁),当客户应用需要更强的锁或新的锁时才向服务器发上锁请求,服务器广播要求其他的客户端归还相应的锁,所有的归还动作结束后服务器才允许请求者上锁成功.

O2PL:客户端的事务可以只上局部锁来更新其缓冲区中的页面,但进入提交阶段前不允许将脏页面传回服务器.根据开始提交时采取的不同动作又可将O2PL分为:O2PL-I(一旦开始提交,服务器将所有持有更新页面的客户端的页面置Invalidate标志);O2PL-P(提交时直接将所有客户端的相应旧页面一次全部更新);O2PL-D(根据不同的环境负担,动态使用O2PL-0I或O2PL-P).

2.2 各种并发控制方法的比较

B2PL由于没有利用事务间的局部性,因而不能很好地胜任客户/服务器的计算环境.在客户数少的情况下,B2PL无优势可言,而且由于其频繁的I/O操作,它的事务吞吐量随着客户数的增多很快达到了最大值并回落.

剩下的几种方法都利用了事务间的局部性,所以比B2PL好,这几种方法其实可大致分为两大类:悲观的和乐观的.C2PL和CB2PL属于前者,而O2PL的几种变型属于乐观的算法.

悲观的并发控制与乐观的并发控制有不同的行为,主要就是在冲突不多的情况下,乐观方法比悲观方法好得多,但事务和冲突一旦频繁,乐观方法性能的下降是非常快的.这里讨论的C2PL,CB2PL与各类O2PL的区别也有这个特点.

在参考文献[10]中,作者测试的结果是在大多数情况下O2PL表现比C2PL好.但是,应注意到,在客户端缓冲25%的数据库页面在大型应用中是不可思议的,而且作者的测试最大客户数只有25个,从性能曲线的走向看,O2PL的性能将继续下降,而C2PL则稳定在一个适中的值.所以,这种测试结果的前提是硬件环境和应用环境足够宽松.对于建立一个实用的DBMS,纯粹乐观的并发控制是不够的.

虽然O2PL的性能在负荷重的环境中下降很快,而且也可以说C2PL的性能稳定得多,但事实是,一直在达到很重的负荷以前,O2PL确实比C2PL性能好得多(这在局部性很强的CAD应用环境中尤其明显).所以,如何向C2PL加入一些"乐观"的特性,使之在负荷不重、局部性好的环境中性能有所提高是值得思考的.CB2PL是其中的一个算法.参考文献[10]没有测试CB2PL,在参考文献[11]中比较了CB2PL与C2PL,结果表明,CB2PL的这种扩充是成功的.

在现实的系统中,不可能一种并发控制在所有应用环境都有出色表现.C2PL和CB2PL是较折衷和稳定的方案.在非常特定的应用环境中,如高配置的小组CAD设计环境、O2PL及其变型是相当有竞争力的.

3 OSCAR的体系结构

OSCAR是我们自主开发的面向对象的工程数据库管理系统.新版本的总体设计目标是简单性、层次性和鲁棒性.我们认为系统的总体性能应该由体系结构来保证,而在系统的实现上不必拘泥于某些性能改进有限而实现代价高昂的方案.

在对各种体系结构经过广泛深入的了解和比较之后,我们决定采用页面服务器的体系结构.这种方案的主要优点是简化了服务器端模块,支持更多的客户数,而且其性能上的优越性在硬件不断发展的情况下会更加显著地体现出来.

OSCAR在服务器端的模块由底向上分为4层:I/O层封装了所有需要的物理文件操作;物理层屏蔽了物理文件的概念;缓冲层屏蔽了内存和外存的区别;事务层提供了基于页面的并发控制和恢复功能,通过网络向客户端提供页面级的数据库访问服务.其中事务层是设计的重点.

我们使用了CB2PL来保证缓冲一致性.我们认为,CB2PL比B2PL增加的实现上的复杂度是有限的(比起在页面服务器上实现对象级锁要简单得多),但性能的提高是显著的.

OSCAR的并发控制粒度是页面.虽然页面锁使并发度有所下降,但在这里换取实现上的简单性是值得的.因为在页面服务器上实现对象级锁不仅会大大增加并发控制模块实现上的复杂性,而且恢复模块也会因此更加庞大.所以,我们在降低锁粒度之外还寻求提高并发度的其他手段,比如对索引页采取多版本的二阶段锁协议(MV2PL)来提高索引页的读写并行能力.

数据库恢复采用了ARIAS[12]策略,这是一种比较成熟的恢复策略.OSCAR的恢复策略扩充了经典的

ARIAS,使之适用于页面服务器的体系结构.我们在一些关键环节对 ARIAS 进行了修改,并论证了扩充方案的正确性.其主要的修改在于分配 LSN 和客户/服务器环境下 WAL(先写日志协议)的实现.日志方法采用基于页面的物理日志,这在服务器端没有模式信息而恢复又在服务器端执行的页面服务器体系结构中是必需的.OSCAR 的恢复策略保留了 ARIAS 原有的一些像模糊检查点(做检查点时只记脏页表)等特性,继承了 ARIAS 在性能上的优越性.

4 结 论

本文论述了设计工程数据库管理系统的体系结构需要考虑的两个重大问题:客户/服务器结构和并发控制策略.介绍和分析比较了 3 种客户/服务器体系结构及传送数据客户/服务器环境中的缓冲一致性问题,也介绍了我们自主开发的工程数据库管理系统 OSCAR 所采取的方案.

参考文献

1 Franklin M *et al*. Crash recovery in client/server EXODUS. In: Stonebraker M ed. Proceedings of the ACM SIGMOD International Conference'92 on Management of Data. San Diego: ACM Press, 1992. 165～174

2 Kemper A, Moerkotte G. Object-oriented Database Management: Applications in Engineering and Computer Science. New Jersey: Prentice-Hall, 1994

3 Deux O *et al*. The story of O2. IEEE Transactions on Knowledge and Data Engineering, 1990,2(1):91～108

4 Object Design Inc.. ObjectStore Tutorial Release 1.1 for UNIX-Based Systems. Burlington: Object Design Inc., 1991

5 Kim W *et al*. The architecture of the ORION next-generation database system. IEEE Transactions on Knowledge and Data Engineering, 1990,2(1):109～124

6 Dewitt D *et al*. Three alternative workstation-server architectures. In: Bancilhon F ed. Building an Object-oriented Database System——The Story of O2. San Mateo: Morgan Kaufmann Publishers, 1992. 411～446

7 Effelsberg W, Haerder T. Principle of database buffer management. ACM Transactions on Database Systems, 1984,9(4):560～595

8 Chou H T, Dewitt D J *et al*. Design and implementation of the wisconsin storage system. Software: Practice and Experience, 1985,15(10):943～962

9 Bernstein P, Hadzilacos V, Goodman N. Concurrency Control and Recovery in Database Systems. Menlo Park: Addison-Wesley, 1987

10 Carey M, Franklin M, Livney M. Data caching tradeoffs in client-server DBMS architecture. In: Proceedings of the ACM SIGMOD International Conference'91 on Management of Data. Denver: ACM Press, 1991. 357～366

11 Wang Y, Rowe L. Cache consistency and concurrency control in a client/server DBMS architecture. In: Clifford J ed. Proceedings of the ACM SIGMOD International Conference'91 on Management of Data. Denver: ACM Press, 1991. 367～376

12 Mohan C, Haderle D *et al*. ARIAS: a transaction recovery method supporting fine-granularity locking and partial rollback using write-ahead logging. ACM Transactions on Database Systems, 1992,17(1):94～162

The Architectures of Engineering Database Management Systems

CHEN Jun SUN Jian-ling DONG Jin-xiang

(*Department of Computer Science and Engineering Zhejiang University Hangzhou* 310027)

(*State Key Laboratory of CAD & CG Zhejiang University Hangzhou* 310027)

Abstract There are two issues involved in designing the architectures of engineering database management systems. The first is which kind of client/server structure is to be employed to make a proper distribution of functions between client and server. And the second is which kind of concurrency control algorithm is to be used to guarantee the consistency of the caching data at clients. Three client/server architectures (i.e. Object-server, Page-server, File server) are dicussed, the differences among them are designated, the advantages and disadvantages of each are pointed out in this paper. The various concurrency control algorithms for cache consistency are also studied. Finally, the architecture of OSCAR, an engineering database management system, is introduced.

Key words Engineering database management system, architecture, cache consistency, concurrency control.