1. **请给出客户/服务器体系结构的描述，并说明二层、三层和多层 C/S 架构的各自特点（ch2 p1）**

体系结构描述：

它将应用系统的计算机分为：

客户机：应用逻辑、表达逻辑

服务器：数据管理、事务逻辑。

C/S 结构把数据处理任务功能分开在客户端和数据库服务器上进行，带来明显好处：

(1) 有利于充分利用网络中的计算资源；

(2) 大大减少了网络上的传输量：只传输数据处理结果

二层：二层 C/S 结构中，每一个客户端需要配置好几层软件，如操作系统、网络协议软件、客户机软件及应用程序等，这样导致 Client 端变得很庞大，故被称为“肥”客户机；

在服务器端则是单纯的数据库服务器，称为“瘦”服务器。

三层：为解决二层 C/S 结构的问题，提出了三层 C/S 结构的网络计算模式。

表示层：表达逻辑(显示与交互)—客户机

功能层：应用逻辑(应用层)—应用服务器(群)

数据层：数据管理,事务逻辑—数据库服务器(群)

多层：多层结构由以下三类分层来定义：

(1) 前端的客户层：负责提供可移植的表达逻辑。

(2) 中间的应用层：实现各类业务逻辑，并可根据实际需要再分解为若干层。

(3) 后端的数据管理与服务层：提供对专门数据服务(如数据库服务器)的访问

多层：多层结构与传统二层结构的区别：两层结构把应用逻辑放置到客户端或数据库服务器。多层结构则将应用逻辑放到单独的中间层上，明确划分表达逻辑、应用逻辑和数据服务逻辑。

三层结构是多层结构的特例。

三个维度：集中数据库、客户端服务器、共享数据库

集中式：主机：运行所有程序(DBMS、应用程序、通信软件等)，还有所有数据处理工作。

终端：负责处理用户的输入、输出等简单功能，

主机 OS 一般是分时系统，用户通过终端与主机上运行的应用程序交互、访问数据库。

早期用户终端没有智能，不具备独立计算能力；现在的终端可用智能终端，或微机代替。

客户端服务器：它将应用系统的计算机分为：

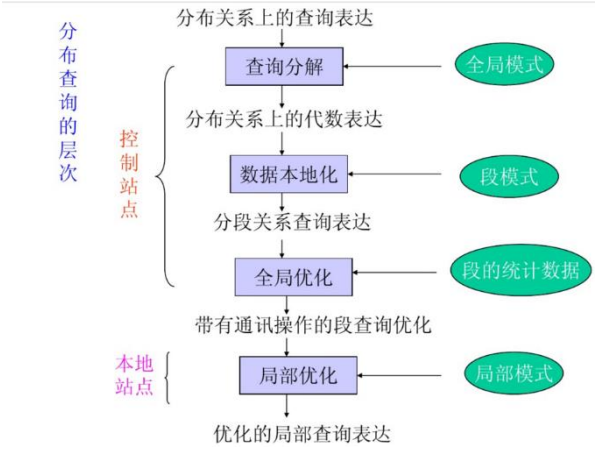
客户机：应用逻辑、表达逻辑

服务器：数据管理、事务逻辑

共享数据库：（自己找）

2. **说明分布式数据库系统 DDBS 中透明性的作用和种类（见下一）**

3. **请详细说明分布式查询处理的过程中以及各阶段完成的任务**



规范化：检查词汇和语法，验证有效性；将语句变成正规形式。

分析：检测和拒绝不正确的查询；可能仅用于关系计算的子集。

简单化：消除冗余的谓词。

重新结构化：关系查询转化为代数查询；可能需要不止一次的转换；使用转换规则。

Distributed Query Processing Page3 最下面两张开始

4. **请说出 ANSI/SPARC 三层架构的特点，并说明分布式数据库和多数据库所做的扩展。（ch4 p1）**

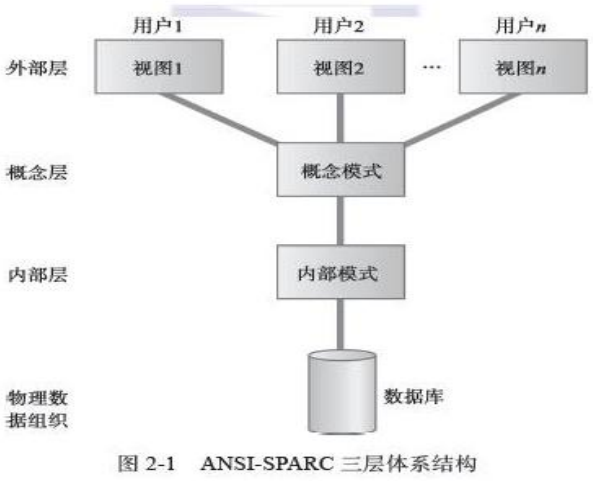


图 2-1 ANSI-SPARC 三层体系结构

外模式（视图）：他提供给用户一个抽象的视图

模式（模式）：DB 的公共视图

内模式（模式）：负责数据的存储

用户从外部层观察数据，DBMS 和操作系统从内部层观察数据。概念层提供内、外部层的映射和必要的独立性。三层体系结构的目的是将用户的数据库视图与数据库的物理描述分离开。如此分离的原因可概括为：

（1）每个用户都能访问相同的数据，但可用各自定制的数据视图。每个用户都应该能够改变自己的数据视图，但这些改变不会影响其他用户。

（2）不要求用户直接处理数据库物理存储的细节，例如索引或散列。换句话说，用户与数据库的交互应该独立于存储细节。

（3）数据库管理员（DBA）能在不影响用户视图的情况下修改数据库存储结构。

数据库的内部结构不受存储的物理变化的影响，例如将数据转存到某个新的存储设备上。

（4）DBA 能在不影响所有用户的情况下修改数据库的概念结构。

扩展：

ANSI/SPARC 中的外模式->分布式数据库中的外模式

ANSI/SPARC 中的模式->分布式数据库中的全局概念模式

ANSI/SPARC 中的内模式->分布式数据库中的本地内部模式

全局概念模式和本地内部模式支持位置和复制的透明度。

网络透明度由全局概念模式支持。

分布式 DBMS 将全局查询转换为一组本地查询。

本地查询由不同站点上的分布式 DBMS 组件执行。

5. **关系模式 R(U,F)， 其中 U = {A,B,C,D,E,I}， F={A→D，AB→E，BI→E，CD→I，E→C}，请按步骤做闭包计算{ACE}+，并说明属性集团包的作用（ch3 p6）**

   1. 令 X={A,C,E}

   2. i.A-D,将 D 加入 X，得 X={A,C,D,E}

      ii.CD->I,将 I 加入 X，得 X={A,C,D,E,I}

      iii.没有属性可以加入 X

   3.得到 X={A,C,D,E,I}

   如果知道怎么计算属性集的闭包，可以测试功能性依赖 A1A1…An->B 是否由依赖集 S 推断而来。

6. **请给出事务的四个特点，并举例说明何为并发事务的可串行化调度（ch10）**

原子性，一致性，隔离性，持久性。

原子性：一个事务是一个不可分割的工作单位，事务中包括的诸操作要么都做，要么都不做；

一致性：一致性是指事务必须使数据库从一个一致性状态变换到另一个一致性状态，也就是说一个事务执行之前和执行之后都必须处于一致性状态；

隔离性：一个事务的执行不能其它事务干扰。即一个事务内部的操作及使用的数据对其它并发事务是隔离的，并发执行的各个事务之间不能互相干扰；

持久性：持久性是指一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性的，即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。

可串行化调度：如果一个调度等价于某个串行调度，则该调度称为可串行化调度。也就是说，该调度可以通过一系列非冲突动作的交换操作使其成为串行调度。比如，两个事务的并发调度的某一执行方式，与先执行 T1 后执行 T2（或先执行 T2 后执行 T1）的结果是一样的，称这一调度符合可串行化调度。

7. **垂直分割中属性紧密矩阵（Arrrubte Affinity matrix）和聚合紧密矩阵（Clustered Affinity matrix）有何作用，请结合具体利息说明垂直分割实现的过程和步骤（ch5）**

   亲和度：垂直划分将被一起访问的属性放在一个片段，衡量如何“在一起”的概念

   属性亲和度矩阵：即属性亲和度的矩阵表示。在数据分片设计中使用，需要穷举属性亲和举证的列排列，行与列同时调整，

   聚类亲和度矩阵：在属性亲和度值矩阵 AA 基础上，找到把一个关系的属性进行分组的方法。键能算法以属性亲和度矩阵作为输入，对行列进行排列，生成的矩阵（CA）

   发现好的“分割点”，才能极大化每个分割内的亲和力，极小化分割的访问

8. **请说明对关系模式进行规范化的过程（ch03）**

规范化理论把关系应满足的规范要求分为几级，满足最低要求的一级叫做第一范式(1NF)，在第一范式的基础上提出了第二范式(2NF)，在第二范式的基础上又提出了第三范式(3NF)，以后又提出了 BCNF 范式，4NF，5NF。范式的等级越高，应满足的约束集条件也越严格。

（1）取原始的 1NF 关系投影，消去非主属性对键的部门函数依赖，从而产生一组 2NF 关系。

（2）取 2NF 关系的投影，消去非主属性对键的传递函数依赖，产生一组 3NF 关系。

（3）取这些 3NF 的投影，消去决定因素不是键的函数依赖。产生一组 BCNF 关系。

（4）取这些 BCNF 关系的投影，消去其中不是函数依赖的非平多值依赖，产生一组 4NF 关系。

5FN 是在第四范式的基础上做的进一步规范化。

4FN 处理的是相互独立的多值情况，而第五范式则处理相互依赖的多值情况。

对于存在数据冗余、插入异常、删除异常问题的关系模式，应采取将一个关系模式分解为多个关系模式的方法进行处理。一个低一级范式的关系模式，通过模式分解可以转换为若干个高一级范式的关系模式，这就是所谓的规范化过程。

9. **请说明分割的正确性包含哪三个方面，以及分割时需要考虑哪些类型的信息。**

完整性：关系 R 分解成分片 R_1，R2…Rn 是完整的当且仅当关系 R 中的每个数据项可以在某个分片 Ri 中找到；

重建性：如果关系 R 倍分解成分片 R1，R2…Rn，那么应该存在某个操作符“倒三角形（DataBase Densign Page2 左下角）”使得。。。

不相交性：如果关系 R 倍分解成分片 R1，R2…Rn，并且数据项 dj 在 Rj 中，那么 dj 不应该存在于其他任何分片中。

信息类型：DataBase Design Page3 右上角开始

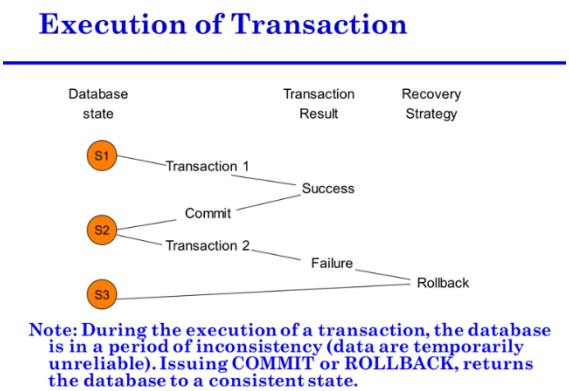
（1）数据库信息（2）应用信息（3）通讯网络信息

（4）计算机系统信息

10. **在恢复时哪种错误类型是在分布式环境下要特别考虑的，分布式事务的执行过程是怎么样的，两阶段提交协议和三阶段提交协议的主要思想是什么？**

分布式环境下需要特别考虑通讯失败。

分布式事务执行过程：



二阶段提交的算法思路可以概括为：参与者将操作成败通知协调者，再由协调者根据所有参与者的反馈情报决定各参与者是否要提交操作还是中止操作。

三阶段提交有两个改动点。

1、引入超时机制。同时在协调者和参与者中都引入超时机制。

2、在第一阶段和第二阶段中插入一个准备阶段。保证了在最后提交阶段之前各参与节点的状态是一致的。

也就是说，除了引入超时机制之外，3PC 把 2PC 的准备阶段再次一分为二，这样三阶段提交就有 CanCommit、PreCommit、DoCommit 三个阶段。

11. 请谈谈对 Hadoop 或 Spark 框架的理解。

解决问题的层面不一样

首先，Hadoop 和 Apache Spark 两者都是大数据框架，但是各自存在的目的不尽相同。Hadoop 实质上更多是一个分布式数据基础设施: 它将巨大的数据集分派到一个由普通计算机组成的集群中的多个节点进行存储，意味着您不需要购买和维护昂贵的服务器硬件，Hadoop 还会索引和跟踪这些数据，让大数据处理和分析效率达到前所未有的高度；Spark，则是那么一个专门用来对那些分布式存储的大数据进行处理的工具，它并不会进行分布式数据的存储。

两者可合可分

Hadoop 除了提供了一个为大家所共识的 HDFS 分布式数据存储功能之外，还提供了叫做 MapReduce 的数据处理功能，所以我们完全可以抛开 Spark，使用 Hadoop 自身的 MapReduce 来完成数据的处理；Spark 也不是非要依附在 Hadoop 身上才能生存，但如上所述，毕竟它没有提供文件管理系统，所以它必须和其他的分布式文件系统进行集成才能运作，这里我们可以选择 Hadoop 的 HDFS,也可以选择其他的基于云的数据系统平台，但 Spark 默认来说还是被用在 Hadoop 上面的，毕竟大家都认为它们的结合是最好的。

Spark 与 MapReduce 是一种相互共生的关系。Hadoop 提供了 Spark 所没有的功能特性，比如分布式文件系统，而 Spark 为需要它的那些数据集提供了实时内存处理。完美的大数据场景正是设计人员当初预想的那样：让 Hadoop 和 Spark 在同一个团队里面协同运行。

1. 请说明分布式数据库系统 DDBS 的承诺，并详细推述透明性的作用。（ch1 page2、3）

分布式数据库系统 DDBS 的承诺：1、分布式透明管理，碎片化和数据复制。2、通过分布式事务提高可靠性/可用性。3、提升性能。4、更容易和更经济的系统扩展。

透明性的作用：将系统的高层语义与低层实现问题分离开来。在分布式环境中提供数据独立性。

透明性的种类：1、网络（分配）的透明性。2、复制的透明性。3、分片的透明性（水平分片：选择、垂直分片：投影、混合）

2. 请给出 DBMS 体系结构模型三个维度的说明，并重点给出客户/服务器体系结构的描述。（见上 1）

3. 请说出水平分割的特点，并说明基本水平分割的步骤和引导水平分割的基本原理（ch5 p6、7）

水平分割的特点： 水平分割会给应用增加复杂度，它通常在查询时需要多个表名，查询所有数据需要 union 操作。在许多数据库应用中，这种复杂性会超过它带来的优点，因为只要索引关键字不大，则在索引用于查询时，表中增加两到三倍数据量，查询时也就增加读一个索引层的磁盘次数。

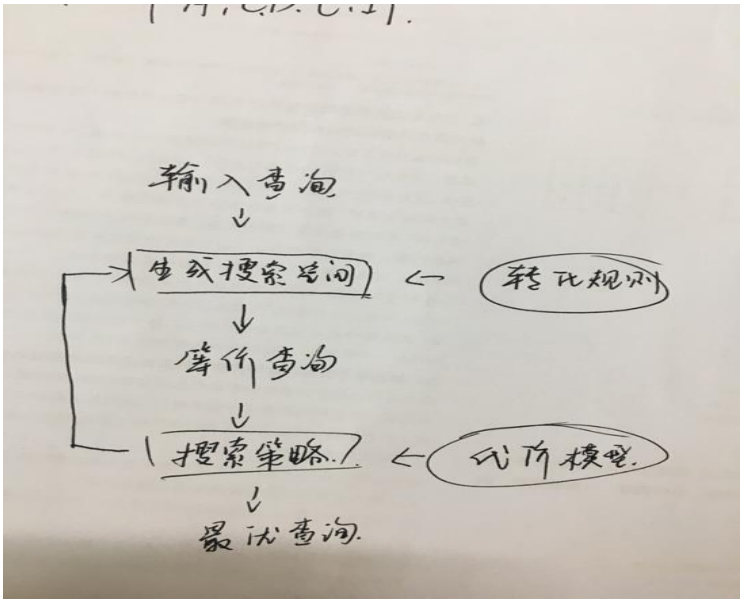
基本水平分割的步骤：使用 COM_MIN 算法执行碎片。输入：关系 R 和一组简单谓词 Pr。输出：根据其关系 R 被分割的一组最小谓词 M。步骤：1、确定最小谓词的集合 M。2、确定 pi∈Pr 中集合 i 的含义。3、从 M 中消除矛盾的最小项。

引导水平分割的基本原理：根据其所有者指定的选择操作在连接的成员关系上定义：每个连接是等值连接，等值连接可以通过半连接实现。

4. 在垂直分割中属性紧密矩阵 Atribute Afininy matrix)和聚合紧密电阵(Clustered afininy marisc)各自代表什么，如何构造，请用下面的数据简单示例说明。(ch5 p10 上 7)

Process	Columns	Affinity
P1	C1,C4	5
P2	C2.C3	2
P3	C2.C4	3
P4	C3.C4	0

5. 请说明分布式查询处理中查询优化是如何实现的，代价模型起到什么作用，并试用典型算法加以举例说明。



代价模型的作用：CH07-09

减少每个要素的时间代价；尽可能少的使用每个成本要素；优化资源利用；增加系统吞吐量。

典型算法举例说明：概念那份的 3.3.

分布式数据库中的查询过程可分为逻辑分解、评议转换和优化组台积分。分布式数据库系统中，用户可以用全局查询评议与多个数据库同时进行查询，即为全局查询。全局查询一般经过以下几个过程：首先，对全局查询进行逻辑分解成几个子查询，每个子查询对应一个局部数据；其次，若全局查询评议与局部查询评议不同，则进行语言的等价转换；最后，各个子查询的结果优化组台后返回。不同的麦询分解对应不同的系统性能，因此为了达到优化系统性能，需要相应查询优化器来确定一个相对较好的执行计划，最后启动查询计划。

6. 请说明等待图的构造和作用，在分布式环境中等待图的表达有什么扩展。

7. 请说明分布式环境下两阶段提交协议为何会出现阻塞，三阶段提交协议有何不同，是如何解决阻塞问题的为何出现阻塞（ch10）

1、同步阻塞问题。执行过程中，所有参与节点都是事务阻塞型的。当参与者占有公共资源时，其他第三方节点访问公共资源不得不处于阻塞状态。

2、单点故障。由于协调者的重要性，一旦协调者发生故障。参与者会一直阻塞下去。尤其在第二阶段，协调者发生故障，那么所有的参与者还都处于锁定事务资源的状态中，而无法继续完成事务操作。(如果是协调者挂掉，可以重新选举一个协调者，但是无法解决因为协调者宕机导致的参与者处于阻塞状态的问题)

三阶段提交有两个改动点。

1、引入超时机制。同时在协调者和参与者中都引入超时机制。

2、在第一阶段和第二阶段中插入一个准备阶段。保证了在最后提交阶段之前各参与节点的状态是一致的。

也就是说，除了引入超时机制之外，3PC 把 2PC 的准备阶段再次一分为二，这样三阶段提交就有 CanCommit、PreCommit、DoCommit 三个阶段。

如何解决阻塞问题的

3PC 主要解决的单点故障问题，并减少阻塞，因为一旦参与者无法及时收到来自协调者的信息之后，他会默认执行 commit。而不会一直持有事务资源并处于阻塞状态。但是这种机制也会导致数据一致性问题，因为，由于网络原因，协调者发送的 abort 响应没有及时被参与者接收到，那么参与者在等待超时之后执行了 commit 操作。这样就和其他接到 abort 命令并执行回滚的参与者之间存在数据不一致的情况。

8. 请给出你所在小组课堂报告题目和主要内容，并根据你所阅读的四篇会议文献，你认为在该方向的研究中目前存在的问题主要有哪些。为什么（见最后）

9. 根据对 HBASE 源码的阅读，请你谈谈对 HBASE 的理解

大数据时代的数据量是超大规模的，传统的关系数据库已经很难存储和管理这些数据了，为了存储海量数据，我们有了 HDFS，它可以把成千上万台服务器上的硬盘聚集成一块超级大的硬盘，为了让这些数据产生价值，我们有了 mapreduce，它可以计算这个超大硬盘的数据，面对这么大的数据量我们还有一个迫切的需求那就是如何快速检索出我们想要的数据，而这个功能就是由 hbase 来承担。

那么如此海量数据快速检索技术原理又是怎样的呢？我觉得原理很简单就是索引技术。Hbase 通过 rowkey 来区别不同类型数据，通过列族把经常需要一起被查询出来的数据放在一起，例如我们如果要做一个电商平台的交易记录业务表设计，对于电商平台下的商户他其实只需要查询出自己的交易信息，而不会去关心其他商户的交易信息，那么我们就可以把商户号作为 rowkey，每一个商户的交易的信息我们就放在一个列族里，商户号这样的信息就像数据在硬盘上的门牌号，我们一传入这个值做查询，hbase 就能快速找到数据存储的位置，这就是 hbase 能快速检索到数据的原理。

上面讲到的原理只是业务抽象的角度来说，在 hbase 底层它就是根据上面说到的这些原理来设计的，hbase 里面有 region 的概念，region 是一个数据集合，那么什么样的数据会放置到某一个 region 里呢？hbase 是根据 rowkey 来把同一类的数据放置在一个 region 里，rowkey 下面就是列族，列族对应的底层存储就是 hfile，hfile 放置在 rowkey 对应的 region 下，所以当我们查询时候我们很容易通过业务规则找到我们设计好的 rowkey，找到了 rowkey 就找到 region，那么 region 下存储的 hfile 列族信息也就可以全部查询出来了。

Rowkey 其实就是 hbase 的索引，也可以说是 hbase 官方给出的唯一索引，因此很多资料里说 hbase 只有一级索引，这个一级索引就指的就是 rowkey，因此如何设计 rowkey 就是一门大学问了，时常我们一行数据不能满足我们复杂的查询要求，我们需要跨行就像 scan 那么扫描多行数据，而 region 里的行都是按照一定顺序排列的，这个顺序就是字典顺序，这个我在以前一篇文章里提到过，所以碰到这种情况，我们一般会通过 md5 将 key 散列，这样相邻的数据行会排列在一起，底层存储数据时候也会存储在同一个地方(相同 region)或者是相互靠近的地方(相邻 region)，这样也就可以提升查询的效率。

Hbase 内部有两张表一个是-ROOT-表和.META.表，客户端程序就是像我上面给出的示例程序首先访问 zookeeper，通过 zookeeper 获取含有-ROOT-的 region 服务器名，通过-ROOT-的 region 服务器可以查询到.META.表里行键 rowkey 对应的 region 位置，而-ROOT-和.META.客户端访问后就会缓存起来。

其实 hbase 的表设计本身非常简单，对外接口也没有关系数据库那么丰富，我最近学习 hbase，觉得 hbase 基本都没有关系数据库里那些计算函数，可见 hbase 只是提供一种能快速检索海量数据的一种计算模型而已。

论文报告

确定偏好空间中竞争选项的影响区域

论文研究所针对的问题和背景：

本篇论文所针对的主要问题为：如何在各种指标数据下，让自己跻身 top-k 中。举个例子，对于一家餐厅来说，顾客选择是否去某家餐厅吃饭时，会综合考虑多个因素，比如价格、环境以及服务，每个顾客在选择某家餐厅时都会根据这些指标对餐厅进行排名，排名位于前 k 个的餐厅会成为顾客的首选。考虑到极限情况，某位顾客是否选择一家餐厅主要取决于这家餐厅的价格，即此时价格指标的权重为 100%，环境以及服务指标的权重为 0%，此时决定该顾客是否选择一家餐厅的唯一因素就是价格。考虑到大部分的情况下，决定顾客是否选择某家餐厅需要综合考虑多个因素的加权和，也就是所谓的得分。得分越高排名就会越靠前，唯一不同的是不同的客户考虑多个因素时对每个因素所分配的权重不同。餐厅方面则会通过提升自己在价格、环境以及服务方面的质量和水平，使自己的餐厅能够在大多数情况下排名都在 top-k 内。

在各行各业中，同行竞争已经变得越来越激烈，想让餐厅甚至企业在同行中占据更高的市场份额，只有让自己的服务在各方面有所提升，而只有在明确的知道顾客想要什么以及了解自己的不足，才可以更加有针对性地提升自己在同行中的排名，吸引更

多的顾客。本片论文的研究意义就在于此，让企业更好地定位自身，提升服务，在大部分情况下都能在同行中排名靠前（进去 top-k 内）。

研究的主要内容：

本篇论文研究的主要内容是确定偏好空间中竞争选项的影响区域，即确定在给定偏好空间中某一个选项在满足排名在 top-k 内的影响区域，进而通过改进一些指标以使其在更大范围内排名靠前（在 top-k 内）。

采用的技术和方法：

本篇论文所采用的的技术方法为：在定义一种新的数据结构（单元树，CellTree）的基础上，构建了所谓的单元树方法。通过将经过超平面分割的偏好空间构造成一棵单元树，通过一些引理将无关的部分排除（以此达到剪枝的目的），从而更快地确定其影响区域。

在基础单元树方法的基础上，通过控制 D 中记录的处理顺序，即它们超平面插入 CellTree 的顺序、忽略不会影响排名结果的记录，即剪枝以及基于关键性观察加速插入算法三个方面对其进行了改进，加快了基础单元树方法的效率。

通过早期修剪没有希望的单元、早期检测属于排名结果的单元以及计算 c 的排名上限以及排名下限以达到更加高效的剪枝，从而进一步提高了算法的效率。

研究工作的创新点：

本篇论文研究的创新点主要在于提出了一种数据结构，加上对高维偏好空间所采取的降维操作，以此简化了实际问题。通过引理证明了操作的有效性的同时，对基本的方法进行了一定程度的优化。剪枝技术的应用以及通过定义排名上下限以此来达到更加高效的剪枝效果。研究的最后结论和成果：

在本篇论文中，作者介绍了 k-Shortlist 偏好区域问题（kSPR），它确定了偏好空间中所有区域的特定选项（焦点记录）排在前 k 个可用选项之间。 该问题适用于市场影响分析和定向广告等。 作者开发了一套技术和数据结构（例如，CellTree，基于 LP 的测试，预见性技术），以有效地解决这个问题。 作者提供了一个关于 NBA 数据集的案例研究来证明 kSPR 的有用性，并且足作者用基准数据集上的实验验证了该方法的效率和实用性。未来工作的一个有趣的方向是近似的 kSPR 算法，具有准确性保证，用于更快的处理。

其他组 paper 相关内容：

第二小组的论文讲的是如何安排反馈感知的社交活动参与者。作者考虑事件参与者安排策略的在线情景，研究了一种针对在线情景的新事件参与者安排策略，即反馈感知社交活动参与者安排（FASEA）问题，其中安排的满意度得分是自适应学习的，用户可以选择接受或拒绝安排的事件。特别是，他们将问题建模为上下文组合设置，并使用有效的算法来解决问题。通过广泛的实验研究评估解决方案的有效性和效率，作者的研究结果表明，据报道汤普森抽样在 FASEA 下表现不佳。

第三组的论文讲的是如何在推荐系统中进行快速且精确的产品检索。作者提出了基于顺序扫描的快速和完全内部产品检索（FEXIPRO）框架，其中包括三个要素。首先，FEXIPRO 对产品 P 进行 SVD 变换，之后的几个维度捕获大部分内部产品。这使他们能够通过仅用 q 计算它们的部分内积来修剪项目向量。其次，他们构造一个 P 的整数近似版本，它可以用来计算能够修剪项目向量的内部产品的快速上界。最后，他们对 P 应用一个无损耗变换，使得得到的矩阵只有正值，从而允许内部产品随维度单调递增。实际数据的实验表明，作者的框架通常比一般数量级的方法更胜一筹。

第四组的论文讲的是如何有效计算遗憾率最小化集，即如何使系统推荐给用户的内容让用户更加满意，以减小用户对推荐结果的遗憾。作为本文的主要结果之一，作者开发了一种近似算法，该算法在线性时间内运行，并保证在距离最佳后悔比率的任意小的用户可控距离内保证遗憾比率。 在综合和公开可用的真实数据集上的全面实验证实了他们提出的算法的效率，输出质量和可扩展性。