网格数据检索中结果集的合并算法

王韵婷 战 疆 王 珊
(中国人民大学信息学院 北京 100872)

摘 要 随着网格从科学计算转到企业级应用,要求数据库提供多种服务支持以实现更强更丰富的资源共享和应用。网格上的数据库只能通过网格服务进行访问,而数据库中的数据也只能通过网格服务接口来存取。因此如何在网格环境下直接对分布在各地的数据库进行高效的检索就是迫切要解决的问题。本文首先提出了一个网格环境下数据检索的体系结构,然后针对该结构下的数值型数据的 Top-k 查询问题给出了 GrangM 算法,它有效解决了来自不同数据源查询结果的合并问题。对该算法的模拟实现表明,它可以快速、高效地合并网格中多结点检索出的结果,减少连接中间结果的大小,降低发送查询请求的通信量。

关键词 网格,数据检索,Top-k 查询,GRangM 算法

Efficient Query Processing in Grid Data Retrieval System

WANG Yun-Ting ZHAN Jiang WANG Shan
(Information School, Renmin University of China, Beijing 100872)

Abstract In many Grid applications, the arrtributes for which users specify target values might be handled by external, auntonomous Grid peer sources with self-interfaces. This paper tries to introduce Top-k query in Grid Data Retrieval System. First, a distributed query processing model for Grid is proposed. Ranking and merging of results are distributed across the Grid Service peers. Next, an efficient Top-k query results merging algorithm (GrangM) is proposed. By the fore-statistical information, the most possible sources are selected to execute the query, so as to greatly improve the search efficiency. Experiments showed that such Top-k query improves the query effectiveness and efficiency.

Keywords Grid, IR, Top-k Query, GrangM

1 引言

自从 1992 年 Ian Foster 首先提出了网格(Grid)的概念[1],关于其基础架构的研究就以惊人的速度发展起来。从早期主要针对科学计算应用的五层沙漏模型架构,到 2002 年 2 月,开放性网格服务架构 OGSA[2,3](Open Grid Services Architecture)及其详细规范 OGSI(Open Grid Services Infrastructure)的提出把 Globus 标准与支持商用的 Web Services 标准结合起来。2004 年 1 月新的网格标准 WSRF(Web Services Resource Framework)草案的发布又进一步把 OGSI 转换成了 6 个用于扩展 Web Services 的规范[5]。至此,网格服务已与商用 Web 服务彻底融为一体了。对于网格方面的研究也从科学计算更多地转向到了企业级应用。

网格试图实现互联网上所有资源的共享和协同工作。在网格上"一切皆服务",网格中的数据库只能通过网格服务进行访问,而各数据库中的数据也只能通过网格服务接口来存取。同时,各网格服务所提供的数据资源、查询功能、访问策略等又是动态变化的,在虚拟组织(VO)[4]中的资源可以随时地加入和撤销,即它的部署也是动态的。因此就需要新的机制来实现网格环境下的分布式查询服务。

当分布在不同节点的数据库作为共享资源在网格上发布服务时,对它的查询就不能简单地像分布式数据库一样处理。分布式数据库有统一的模式,因此分布式数据库管理系统可以根据这些模式用 SQL 语句来对数据库中的数据进行查询。但网格上的数据库属于不同的组织和个人,建立于不同的时间,建立时没有统一的规划,因而也不可能有统一的模式。另外网格中的各个结点具有很强的自治性,数据的发布内容、质量和访问方式也由数据所有者控制。这就促使我们研究针对网格环境下的数据检索方法。

当在网格环境下进行信息检索时,用户往往关心的只是最接近其查询条件的那部分结果。于是我们希望使用网格信息检索系统也能够像使用 Web 搜索引擎一样方便,每次查询它都返回匹配度最高的 Top-k 个结果(降序排列与查询条件最接近的 K 个结果)。因此在网格信息检索系统中进行 Top-k 查询是十分必要的,这尤其在电子商务和企业应用领域有着广泛的应用。它在节约网络带宽的同时也提高了用户的满意度。

传统数据库的查询都是精确匹配的查询,不存在近似查找,在关系数据库上进行 Top-k 查询是最近才出现的新的研究方向。文[12]提出了对关系数据库检索的 StopAfter 的实现,文[13,14]在此基础上进一步研究了面向关系数据库的 top-k 检索机制。Fagin 等人提出了 FA (Fagin algorithm), TA (Threshold algorithm)算法[15,16]来处理多属性的 Top-k 查询问题。但当数据库查询作为服务在网格上发布时,数据可被访问的方式由发布者控制,我们只能按照这些接口提供的方式来访问数据,许多数据源只提供及其有限的访问接口,因此就不能简单地将上述算法直接应用到网格环境中,需要针对数据库服务不同的特点,采用不同的技术。

用户提交的多关键词查询条件,分别由网格中不同的数据库服务提供,为了返回与条件最匹配的 Top-k 个结果,我们赋予每个属性一个权重分数(可以根据不同的应用采用多种权重分配)。查询时先将"精确"的请求扩展为它周围一个恰

王韵婷 硕士研究生,主要研究方向:网络数据库与信息检索。战 疆 博士研究生,主要研究方向:数据库、网格和信息人索。王 珊 教授,博士生导师,主要研究方向:数据库与知识工程。

当的范围条件发送给那些可以提供范围查找的数据库服务(比如汽车生产厂家 VO 允许查询某一价格范围内的汽车型号),然后根据合并到的结果属性,搜索其他数据源节点,得到其余属性的值及其分数,最后通过聚集函数综合出的总分,从高到低返回前 K 个结果。

本文第 2 节介绍了网格环境下检索机制的体系结构、查询过程和需要解决的关键问题;第 3 节详细描述了其中的结果合并算法并在第 4 节结合实验数据对其进行了评价;最后总结了全文,并指出了将来的工作方向。

2 系统结构

在 Grid 环境下进行信息检索是在 OGSA 的网格计算平台上实现 IR 的一个新的特殊结构。尽管网格支持的开放标准 OGSA[2,3]统一了接口标准,在一定程度上可以使得用户使用与资源管理分开,但在实际应用中,还是面临着许多问题。

我们借鉴了 GridIR 工作组的设计思想[7~10],按照传统的 IR 模型将整个网格数据检索体系结构划分为四部分:提交查询的用户结点、提供各种原始资源的虚拟组织/结点、查询分派器结点和索引机结点。它们分别由 OGSA 网格服务来描述,同时整个体系中的所有部件都可以并行的运行。

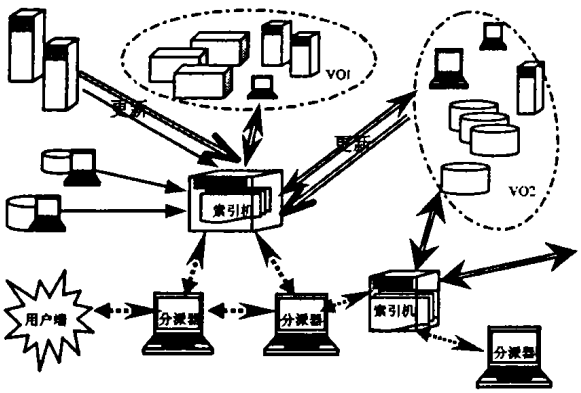


图 1 网格数据检索系统体系结构

当在网格环境下进行信息检索时,在用户端提交的查询会首先被送到查询分派器。查询分派器主要致力于分布式的搜索,同时提供的查询处理服务负责对提交的查询请求进行条件解析、扩展功能(例如辞典)的调用等。然后将处理后的查询条件分派给它连接的一个或多个索引机或其他的查询分派器。于是就可以通过创建查询分派器来动态地建立分布式搜索网络。这就将传统的分布式查找的静态模型(每个都是通过一个固定的入口点查找)和 Z39.50 模型[11](对分布查找的所有责任都交给最终用户)灵活地协调了起来。

索引机主要提供索引与发现服务,它负责创建索引,跟踪监视各个源结点信息的更新记录(比如可以通过 push 或 pull 结构的事件触发重索引),并对收到的查询请求再使用后台 search 引擎实现对各分布式结点的查找,合并来自多个源的结果返回。网格环境下的索引机功能是和传统的 IR 标准不一样的。它是整个系统中的最关键部件。它的最重要特色就是可以对多种结点源进行分布式的查找,同时进行一些预处理和标准化工作。

网格上各个结点/虚拟组织提供的服务都会按照统一标准进行注册、发布,因此可以很容易地知道在哪种访问权限内都有哪些服务、它们提供什么样的查询接口、提供这些服务结点源的 URL 等信息。根据这些预知信息,可以在索引机上为各数据源结点维护元数据信息或对其中数据的分布进行预统计,这样在查询的时候就可以先根据这些信息对查询条件进行筛选,从而减少中间结果的大小,提高效率。

实现这一体系结构,有许多需要解决的关键技术,如:

1. 各个结点源(数据源、索引机结点、查询分派器结点)之间上下需要提供统一的 API 服务接口和服务描述。2. 因为网格中的各个资源具有动态性,在索引机中就需要考虑如何动态地为各服务结点维护相关信息。3. 用户发出的查询会在网格中进行分布式的查找。那么如何有效地利用查询分派器发散查询请求,实现更大范围的分布式查询就是另一个问题。4. 在索引机端如何对分布在各个结点各个数据库上的表进行快速有效的连接。发散到各处的查询结果如何有效地合并返回。下面,针对其中第四点即查询结果合并问题,进行较深入的研究,提出了比较有效的解决方法,并通过实验对它们的结果进行了评价。

3 查询实现

当在网格数据检索系统中进行 Top-k 查询时,我们没有必要把所有从各个结点收集的局部查询结果都返回,缓存到查询处理结点[6,8,9]然后等待统一处理。下面我们就如何有效解决该问题做出较详细的描述。

3.1 网格环境下的 Top-k 查询模型

在我们的模型中,我们定义查询对象的概念,每个查询对象 R 由一系列属性组成 $A_0, A_1, \cdots, A_n$,这些属性的值由不同的数据源提供(通过调用各自的网格服务)。

定义了查询对象,下面我们定义网格环境下的 Top-k 查询。对分布在网格中的各个关系数据库系统,Top-k 查询就是搜索与各查询目标值最匹配的 Top-k 个结果——一组有序的元组集合,它们的顺序由每个结果与查询目标值的距离远近决定。

每个 R 上的 TOP-k 查询 q 为每个属性指定一个目标值 $\{A_0=q_0, A_1=q_1, \cdots, A_n=q_n\}$。对于任一对象 $t=\{A_0=t_0, A_1=t_1, \cdots, A_n=t_n\}$,我们可以通过打分函数计算 t 对于 q 的分数。特别地,对于每个属性 A_i 都有一个打分函数 $ScoreA_i$ 来计算 q_i 与 t_i 之间的分数 $s_i=ScoreA_i(q_i, t_i)$。为了把各个属性上的分数综合起来,我们为每个属性 A_i 指定一个权重 w_i。综上所述,一个对象 t 对于给定的查询 q 的分数为:

$$Score(q,t)=ScoreComb(s_0,s_1,\cdots,s_n)=\sum_{i=0}^{n} w_i * s_i$$

其中 $s_i=Score_{K_i}(q.t_i)$,而返回的 TOP-k 个结果就是 $Score$ 值最高的 K 个对象。

另外,我们定义网格数据检索系统中数据源服务的两种访问接口:

接口 1—范围访问接口(Range Access,RA 源),它提供如下的服务接口:给定一个属性 K_i 上的查询$[q_l, q_h]$,我们可通过调用该接口服务得到这一范围内的对象列表。例如可通过 RA 源查询某一价格范围内的汽车型号列表。

接口 2—唯一访问接口(OneInOneOut Access,OA 源),该接口只提供如下服务:给定一个对象,该接口只返回该对象在该接口上的某些属性值。对于这种 OA 源,我们只能通过向它提交某个汽车型号,得到该型号汽车的某些特性值,如每百公里耗油量等。

我们的查询算法可分为两大步:首先我们根据某个属性 A_0 生成候选的查询对象,并将它们按照查询排序;然后依次对探测查询对象的其他属性并打分,输出最终结果。

我们的算法基于以下假设:1. 有多个网格节点提供属性 A_0 上的查询结果,因此需要对这些结果合并。2. 这些提供属

性 A_0 上的查询结果的节点均为 RA 源。3. 其余每个属性均只从一个节点上获得。

3.2 索引机中统计表的构造

这里我们假设各个结点在注册、发布服务后，在索引机中对各个结点表数值进行分析，我们希望为数值型查询属性的值域做预划分处理，存储到索引机的统计表中。这样在它收到提交的 Top 查询时，可先将它们扩展分派到这些区间上，再将区间范围作为条件发送给各个结点进行查询。

因为我们关心的只是最终 TopK 个结果，应用这种预处理方法，只对各个结点中最有可能满足条件的某一范围结果做操作，就避免了对各个结点的整张表做连接，大大缩减了中间结果的大小。同时，它也减少了常规方法中要执行的查询数量，进而减少了向各结点发送查询请求的通信代价。

在预建立统计表时，有两种实现划分方法：一种就是等宽划分(E-Width)，即指定每个区间的范围，记录落到该区间中的元组数；另一种是等高划分(E-Depth)，即指定每个区间中的元组数，记录该区间的范围。对于第一种划分，适合值域是平均分布的属性。但对于分布不均匀的属性，可能有些区间中的元组数过多，远大于 TopK 的个数，这就不利于我们进一步调整范围查找的大小。而第二种方法对分布不均匀的属性一样有效，尤其符合我们要做的 TopK 查询。所以这里我们用第二种方法构造统计表。

3.3 GRangM 算法描述

3.3.1 RA 源结果合并

1. 对查询分派器送来的查询关键字 K_1，K_2，…，K_M；及 Top-k 值，在索引机中发现要去哪些结点查找跟这些关键字相关的属性。例如 K_i 可能对应结点 RT_1，RT_2 中的表。

2. 在索引机与该结点对应的统计表 $RTATST_1$，$RTATST_2$ 中，查找该关键字落在相关属性值的哪一个区间(如果统计表划分的区间中元组数< TopK 值，向两边扩展区间)。例如关键字 K_p 落在相关表 $RT1$，$RT2$ 的范围 $R_i^1[L_i^1, L_{i+1}^1]$；$R_j^2[L_j^2, L_{j+1}^2]$中。

3. 进一步调整以 K_p 为中心要发出的查询范围，即比较 K_p 与 R_i^1，R_j^2 的中心 M_i^1，M_j^2 相对位置，调整 r 值：

if $(K_p<M_i^1)$ $r^1=\mathrm{MAX}\{K_p-M_i^1,(K_p-L_i^1)\}$

if $(K_p>M_i^1)$ $r^1=\mathrm{MAX}\{K_p-M_i^1,(L_{i+1}^1-K_p)\}$

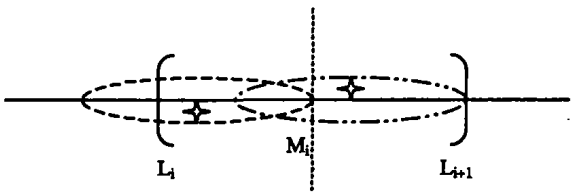


图 2 中括号代表在统计表中预划分的区间，根据提交的查询条件(星号)，我们扩展、调整向各个结点待发送查询范围(用虚线的椭圆表示)。

4. 将范围查询$[K_p-r^1,K_p+r^1]$;$[K_p-r^2,K_p+r^2]$提交给该表对应的结点，然后在每一个结点源上并行地做范围查找。

5. 将结果返回给索引机结点，该结点合并各部分结果，并按照相应的属性列分值降序排列在临时表 $RTEMP$ 中。

这里，结果属性列的分值，实际上就是计算各结果与查询值的相似度数，它保证距离该关键字 K_p 越近，分数越高。在这方面，有很多经典的方法可以借鉴。算法中可以采用各种打分函数，我们在实验中采用了最简单的打分函数：元组 t 与查询 q 的相似度为：

$Score_{t,q}=1-|t-q|/q$

注：这里我们描述的算法是以两个同类结点为例，可以很容易地推广到多个结点的实现。

3.3.2 探测 OA 源

到目前为止我们得到了候选的查询对象表 RTEMP，其中每一个对象都在 A_0 属性上对查询 q 进行了排序。可以从索引结点 $RTEMP$ 表中用 $getnext()$方法顺序得到下一个对象 t(属性 A_0，分数 s_0)，下面我们顺序以每个对象 t 探测各 OA 源。

这里首先定义一个对象 t 的上限 $U(t)$的概念：

从 $RTEMP$ 提取出来的对象 t 的分数上限 $U(t)=Score\text{-}Comb(s_0,s_1',s_2',\cdots s_n')$。其中，若 s_i 已从 OA_i 中获得，则 $s_i'=s_i$；否则 $s_i'=1$。

我们的算法基于以下性质： 假设在某一时刻我们已经从 $RTEMP$ 提取了一些对象，并且已在一些 OA 源上探测了这些对象。如果 $U(t_m)>U(t')$，$\forall\ t'\neq t\in Objects(RTEMP)$，$t_m\in Objects(RTEMP)$，即 t_m 在已探测到的查询对象的中拥有最大的 $U(t)$值，那么，在返回查询结果之前必须还要对 t_m 进行至少一次探测。因为：

· 如果 t_m 是最终 TOP-k 个结果之一，那么我们必须探测它的所有属性以得到它的最终结果。

· 如果 t_m 不是 TOP-k 个结果之一，因为 $U(t_m)$是探测到的对象中的最大值，我们必须进一步探测(使其上限值减少)，以把它排除出 TOP-k 个最终结果之外。

基于上面的性质，我们总是选择 $U(t)$值最大的对象 t_m 进行下一步探测，一直到输出最终结果为止。

具体算法如下：

1. 设 U_{un}记录未被探测到的对象的最大值，Candidates 为已探测到的对象的集合，returned 记录已返回的最终结果的数目。初始化：$U_{un}=U(t)$，Candidates={t}，returened=0。
2. If(Candidates 非空)
   从Candidates 中选取对象 tnow，使 $U(tnow)=\max_{t\in Candidates}U(t')$
   Else：tnow=undefined
3. if (tnow)=undefined or $U(tnow)<(U_{un})$
   (t,s0)←r. getnext()：//从 RA 源中得到下一个对象 t 及其在 A0 上的属性值
   置 $U_{un}=U(t)$，并且将对象 t 插入集合 Candidates；
   Else If(tnow 已经被完全探测)
   返回对象 tnow 及其分数；
   从集合 Candidates 中删除 tnow。(这时 tnow 为最终 TOP-K 个结果之一)
   returned＋＋；
   If(returned=k)结束 //已得到 TOP-K 个结果
   Else
   O_i ←$SelectOASource(tnow)$
   //选取对象 tnow 的下一个未被探测的 OA 源
   s_i←O_i. $getScore(tnow)$
   //在 OA 源上探测对象 tnow
   if (对象 tnow 在 Oi 上返回的值为空)
   从集合 Candidates 中删除 tnow
   //即 Oi 上查不到对象 tnow 的相应属性值
4. 返回第2步

注：其中函数 $SelectOASource(tnow)$在 $tnow$ 未被探测的 OA 源中选取权重最高的返回。最外，所有的 $Score_{t,q}$的值落在[0,1]区间。

4 实验分析

在这一部分，我们实现了上面描述的算法，并在模拟的环境中进行了实验。实验设定有3个 RA 源，5个 OA 源。每个源分处不同的数据库中(即模拟网格中不同结点的数据库)，我们进行的就是对这几个不同数据库中表的查找。这里，假设各个数据库中的表属性没有语义障碍(在这一方面已经有很多相关的模式匹配、语义网络等研究工作可以解决它)，而我们工作的重点是在这之后对查询的结果进行 Top-k 选择。实验中各个结点的数据都是随机产生，并让它们的数据分布不均匀(这样更符合真实环境的状况)，随机在用户端产生查询。下面我们给出网格数据检索系统中 Top-k 查询算法的性能分

析。

4.1 网络通讯代价

主要以用户发出的查询在网格中需要进一步向各数据源结点发送的请求查询数来衡量网络的通讯代价。首先，用我们设计的体系机制，根据在索引机上的统计信息，缩小了模糊查找的范围，以该范围为查询条件发送给各 RA 源(不再以原始的查询请求为条件)，合并它们的结果并按期望排序作为下一步查询的候选元组集(RTEMP)。这就避免了一次次对各 RA 源发查询请求的通讯代价。然后再以这个候选表为出发点，对各 OA 源依其权重分别探测。

解决这类问题常规方法(Normal)是：对候选集(RTEMP)中所有对象的每个属性分别进行探测，按打分函数计算出每个对象的总分排序最后 TOP-K 个结果(如图3)。这种方法，我们必须计算每个对象所有属性的分数，为此不得不反复发送查询请求给所有数据源，也就是对于候选集中的每一个对象，都必须探测所有的 OA 源。假设有 n 个 OA 源，其代价就为 $|Object(RTEMP)|(n+1)$。

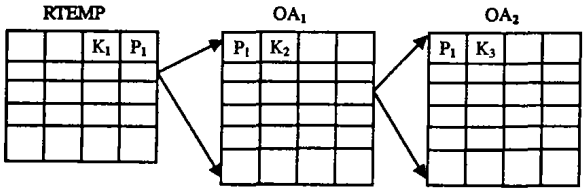


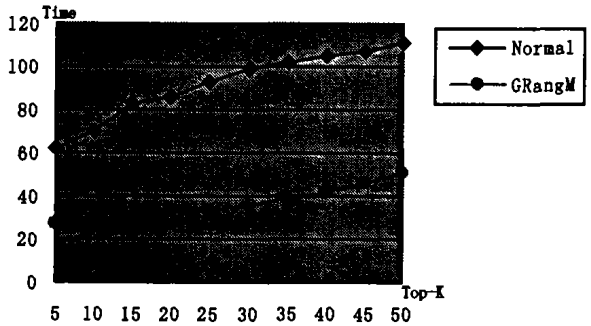
图3 Normal 法探测

这种代价是很高的。而在我们的算法中，我们只访问为得到 TOP-k 个结果所须访问的那些对象，不对侯选集中的所有查询对象探测所有 OA 源，这使得探测数据源的次数大量减少，因而减少了网络通信代价，提高了查询速度。

4.2 查询响应时间

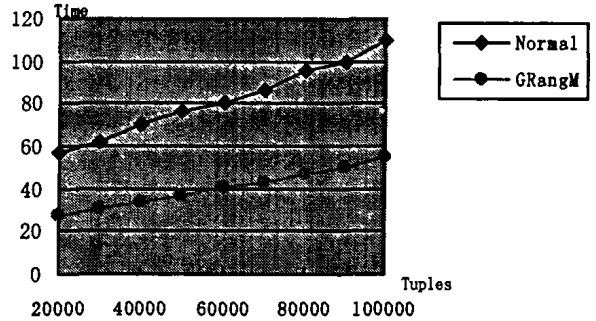
下面我们分别调整不同的参数值，以查询响应的总时间来比较我们的算法与常规算法的效率：

1. *用户要求的 Top-k 值的影响* 当 K 值增加时，我们可以看到，到最终返回 K 个结果，两种方法所需要的总时间都会增加，但我们的算法保持一个更低更平稳的曲线。



2. *结点元组数的影响* 当元组数增加时，两种算法的性能都会降低。每种算法花费的时间都近似的是元组数的线形函数。

结论与展望 本文首先介绍了一个网格环境下数据检索的体系结构、进行Top-k查询的过程及其要解决的主要问题，然后特别针对其中的查询结果合并问题，给出了几个关键技术，并用实验对它们的结果进行了评价。



这里，我们将网格中的数据源结点与索引机结点、查询分派器结点分开，这样就可以允许在网格中进行更大范围的分布式查询收集。另外，我们给出了对来自不同结点数据源的查询结果集合并、权重排序机制。应用这一机制，可以在同样的数据集上采用多种算法和 IR 方法，然后按提供的合并和排序方法来决定合并的结果集的整体排序。与以前的相关研究相比，我们的实现机制有效地减少了中间结果和通信量的大小。

我们在今后的研究中还会对上述网格检索模型逐步递增地进行改进和扩展。对索引机中的数据库服务发现机制，资源定位机制，索引维护机制，及网格中各结点数据库源动态部属时所涉及到的问题进行进一步的研究。

参 考 文 献

1 Ian F, Carl K, Jeffrey N, et al. The hysiology of the Grid
2 Ian F, Carl K, Jeffrey N, et al. The physiology of the grid: An open grid services architecture for distributed systems integration
3 Open GRID Service Architecture (OGSA). http://www.ggf.org/ogsa-wg/
4 Ian F, Carl K, Steven T. The anatomy of the grid: Enabling scalable virtual organizations
5 http://www.globus.org/wsrf/
6 Dovey M. Music GRID - A Collaborative Virtual Organization for Music Information Retrieval Collaboration and Evaluation. The MIR/MDL Evaluation Project White Paper
7 GridIR Working Group Home. http://www.gridir.org/
8 Dovey M J, Gamiel K. GRID IR —— GRID Information Retrieval. Poster at EuroWeb 2002
9 Nassar N, Etymon G, et al. Grid Information Retrieval Architecture. Global Grid Forum(2003). http://www.gridforum.org
10 Gamiel K, Karimi C S, et al. Grid Information Retrieval Requirements. Global Grid Forum(2003). http://www.gridforum.org
11 American National Standards Institute (ANSI). ANSI/NISO Z39.50-1995: Information Retrieval (Z39.50): Application Service Definition and Protocol Specification. Z39.50 Maintenance Agency, July 1995
12 Carey M J, Kossmann D. On saying "Enough already!" in SQL. In: SIGMOD, Tucson, Arizona, May 1997
13 Chaudhuri S, Gravano L. Evaluating top-k selection queries. In VLDB, 1999
14 Donjerkovic D, Ramakr Ishnan R. Probabilistic optimization of top n queries. In VLDB, 1999
15 Fagin R. Combining fuzzy information: an overview. ACM SIGMOD Record, 2002,31(2):109～118
16 Fagin R, Lotem A, Naor M. Optimal aggregation algorithms for middleware. In PODS, Santa Barbara, California, May 2001