文章编号:1004 - 4736(2008)02 - 0094 - 04

一个基于消息通知的工作流管理系统

刘黎志,刘　军
(武汉工程大学计算机科学与工程学院,湖北 武汉 430074)

摘　要:介绍了一个基于消息通知的工作流管理系统,该工作流管理系统以消息通知机制为核心,以消息通知推动流程的运转.工作流模式定义及工作流引擎的设计都围绕消息通知机制进行.用任务表示流程中某个节点中所处理的表单实例,使得表单实例在各个节点的处理过程可以用任务状态表示,工作流引擎通过定时扫描正在处理的各任务状态,控制流程按设定的模式运行.

关键词:消息通知机制;工作流模式定义;工作流引擎
中图分类号:TP 393.07　　文献标识码:A

0　引　言

基于 Internet/Intranet 的应用是目前工作流技术的发展趋势,面向 Web 的工作流管理系统为企业间的相互合作及电子商务提供了有力的支持.企业对工作流管理系统的要求是系统要能合理的、高效的支撑企业业务流程的运转,从而提高企业的生产效率,减少企业的资源消耗[1].下面介绍一个基于消息通知的工作流管理系统,其原型已在湖北清江水电公司的 AMS(Asset Management System)系统中得到实现.

1　工作流管理系统体系结构

工作流管理系统由三个核心模块组成:消息通知机制、工作流模式定义及工作流引擎.工作流定义模块提供用户定义流程的工具并将用户所定义的流程以数据表的形式存储[2].工作流引擎解释工作流定义,并负责流程的流向控制及向任务表写入任务.消息通知机制是系统的核心,消息通知机制定时扫描任务表,将任务表中待处理任务分发到流程中相应的节点处理,以此推动流程的运转.系统体系结构如图 1 所示.

规定表单与流程之间是一一对应的关系,在系统初始化时确定表单的实际载体(WebForm 页面),用任务表示在流程的某个节点中所处理的表单实例.

2　工作流模式定义

工作流模式定义负责将现实世界中业务流程翻译成为计算机能够理解的形式.将节点和流向

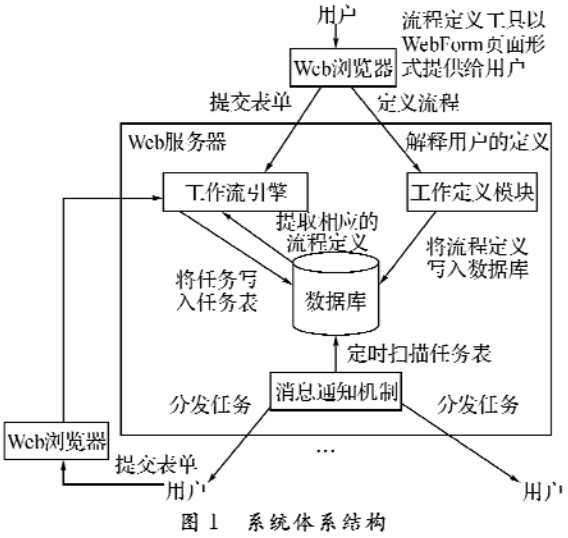

图 1　系统体系结构

Fig. 1　System architecture

控制作为工作流模式的基本要素,可以认为工作流是一个有向图,节点就是图中的点,流程控制是连接点的边.工作流模式定义存放在数据表中,由工作流引擎负责解释工作流模式定义.

2.1　节点

节点是参与流程的用户及角色的抽象表示.表单实例流入某一节点后,节点一般对表单实例进行查看→决策→填写→发送处理,然后由工作流引擎将表单实例送入下一节点,直到流程结束.对节点对象做出以下定义.

nodeId:节点 ID.

nodeName:节点名称.

dscription:节点描述.

type:节点类型.分为四种:Start(开始节点),Synchronize(同步节点),End(结束节点),Handle(处理节点).

收稿日期:2007 - 04 - 12

作者简介:刘黎志(1973 -),男,湖北武汉人,讲师,硕士.研究方向:商业智能、数据仓库及数据挖掘,基于网络的存储系统.

users：节点的用户和角色. 表示参与流程的用户及角色.

isUsed：节点是否已被流程使用. 规定节点只能被一个流程使用一次.

synTimes：任务在节点等待同步结束的次数. 只在节点类型为‘同步’时设置.

2.2　流向控制

工作流模式支持以下几种基本流向控制.

(1) 顺序(Sequence)—— 顺序将表单实例从上一节点交给下一节点.

(2) 并行分叉(And-Split) ——将表单实例同时分发给多个节点.

(3) 并行合并(And-Join)——同步合并多个并行执行的表单实例，直到所有的节点都完成该表单实例，表单实例才流向下一个节点.

(4) 选择分叉(XOR Split)——从一个表单实例的多个流动路径中选择一条符合指定条件的路径，并将表单实例交给该路径连接的节点.

(5) 选择合并(XOR-And)——合并多个排它选择的执行路径，并将表单实例交给下一个节点.

几种基本流向控制如图 2 所示.

图 2　流向控制

Fig. 2　Flow direction control

用一个流程实例说明工作流模式定义，实例的示意图如图 3 所示.

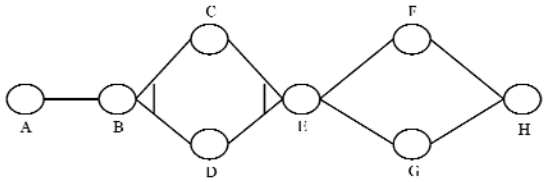

图 3　实例示意图

Fig. 3　Example

其中 A 为‘开始’节点，H 为‘结束’节点，E 为‘同步’节点，其它为‘处理’节点. 该实例的模式定义在数据表中表示如表 1 所示.

表 1　流程实例定义在数据表中的表示

Table 1　A workflow definition in data table

flowId	sourceNode	destinationNode	controlType	condition
Fw001	A	B	Sequence	无
Fw001	B	C	And Split	无
Fw001	B	D	And-split	无
Fw001	C	E	And-Join	无
Fw001	D	E	And-Join	无
Fw001	E	F	XOR-Split	条件 1
Fw001	F	H	XOR-And	无
Fw001	G	H	XOR And	无

3　工作流引擎

工作流引擎的主要功能是根据工作流模式定义决定某个表单实例从一个节点流出后，其所要流向的下一个节点，即工作流的流向控制. 下一节点决定后，工作流引擎负责向任务表中加入任务[3]，使表单实例在下一节点得到处理.

流程流向控制及写入任务的逻辑描述为：

```
/*根据流程 ID 和源节点，得到源节点的目的节点*/
Function string GetDstNode(string flowId,string sNode)
{
    results=在流程定义表中查询流程 ID 为 flowId，源节点为 sNode 的所有记录；
    for(j=0;j<results.length;j++)
    {
        if(result[j].controltype==XOR Split)
        {
            if(result[j].condition 满足)
            {return result[j].destinationNode;
                    break;}
        }
        else
        {return result[j].destinationNode;}
    }
}
/*控制流程的流向，并向任务表中写入任务*/
Function void ExeFlow(string formInstanceId,
string formId,string sNode)
{
    flowId=根据表单与流程的映射关系得到该表单的流程 ID;
    /*得到目的节点对象*/
    dstNode=GetDstNode(flowId,sNode);
    if(dstNode.type==synchronize)
    {
        record=查询任务表中表单实例 ID 为 formInstanceId，表单类别 ID 为 formId，目的节点为 dstNode 的记录；
        /*任务同步的处理*/
        if(record==null)
        {
        写入任务，设置任务记录的 synTimes 字段的值为 dstNode.synTimes-1，record.state 为 wait；
```

```
        }
        else
        {
            record.synTimes--;
            if(record.synTimes==0)
            {设置 record.state 为 needHandle;
            }
        }
    }
    else
        写入任务,设置任务状态为 needHandle;
}
```

4 消息通知机制

消息通知机制的功能是将任务表中的任务状态为'等待处理'的任务分发到相应的节点处理,与有些工作流管理系统使用E-Mail作为消息通知的方式不同,使用基于Web的"推"方式实现消息通知[4,5].

消息通知机制的逻辑由WebForm页面的后台代码执行,该页面按用户所指定的时间间隔定时发送(submit)回服务器端.在服务器端执行的页面后台代码扫描任务表中任务状态为'等待处理'的所有任务.若某个任务的目的节点包含当前用户,则生成一条消息,消息的内容可从该任务记录中提取,并将该消息写入页面的列表域中.页面在服务器端处理结束后,返回到用户客户端的浏览器中.每条消息以超链接的形式展示,引导用户进入承载任务的页面.用户结束任务的处理后,将任务表中该任务的状态设为'处理完毕',并调用工作流引擎的ExeFlow方法,将表单实例送入下一节点处理.当任务在'结束'节点处理完毕后,表示表单实例处理过程的所有任务记录从任务表中删除.

用户成功登录到系统后,用户ID以Session方式保存在Web服务器端.因此该用户ID在其客户端浏览器与Web服务器会话期间始终保持有效,这就可以保证用户在其工作期间得到应该处理的所有任务.

4.1 任务

任务表示在流程的某个节点中所处理的表单实例,任务由工作流引擎写入任务表,由消息通知机制将任务发送给流程中的相应节点处理.任务的格式为:

formInstanceID:表单的实例ID,表示在某一类表单的某一个具体的表单实例.

formId:表单类别ID.

sNode:任务的源节点.

dstNode:任务的目的节点.

flowId:记录表单所在的流程ID.

state:任务的状态.规定任务的四种状态needHandle(等待处理),wait(等待),handling(正在处理),handled(处理完毕)

synTimes:记录任务需要等待的次数.用于并行合并时保证任务的同步.

commitMan:任务的处理人.

applyTime:任务的处理时间.

用任务状态表示任务在处理过程中的各种情况.

4.2 任务的状态

由于处在'同步'节点的任务必须等到所有的分发任务处理完毕后,该任务才能得到处理.因而任务进入'同步'节点后,将其状态设为'等待',待最后一个分发任务处理完毕后,任务状态由'等待'变迁到'等待处理'.

由于一个节点中可以包括多个用户和角色,因此消息通知机制会将在该节点未处理的任务通知给节点中的每个用户以及属于节点所包括的角色的所有用户.如果不加控制,任务可能会被多个用户同时处理,从而出现数据不一致的情况.规定用户开始处理任务时,只有当任务状态为'等待处理'时,才将任务交给用户处理.因此最先收到消息的用户只要决定开始处理任务,就将该任务的状态设定为'正在处理',从而锁定该任务.若用户处理任务完毕,则将任务的状态设定为'处理完毕'.若用户取消处理该任务,则将任务的状态设定为'等待处理',让其他用户有机会处理该任务.

任务表中的某些任务可能始终处于'正在处理'状态.例如,某个用户正在处理表单,此时任务的状态为'正在处理',但在用户提交或取消表单处理前,用户的客户端死机.这就会导致该任务始终处于'正在处理'状态,但却又没有人正在处理该任务.解决这个问题的方法是在服务器端设置一个定时处理程序,根据实际情况定期将任务表中所有的状态为'正在处理'的任务的状态重新设置为'等待处理'.任务状态变迁的可能方式如图4所示.

消息通知使得用户完全在系统的控制下处理任务,但也会在一定程度上影响系统的灵活性,特别体现在对紧急任务的处理上.因为消息通知机制是在一定的时间间隔后才实现消息通知,所以紧急任务的处理需要人为干预.

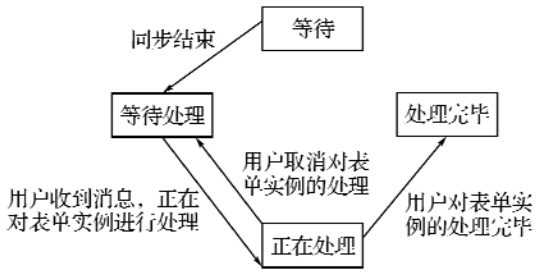

图 4　任务状态变迁方式

Fig. 4　Task state transition

5　结　语

上述工作流管理系统原型已经在 AMS 系统中得到实际应用，基本上可以达到预期效果. 消息通知是设计该系统的核心，业务流程完全靠消息通知推动完成. 如何提高消息通知的灵活性，实现系统在分布、异构环境下的可适应性、扩展性及对不确定型工作流的控制将是今后研究的方向.

参考文献：

[1] Hollingsworth D. The Workflow Reference Model [M]. Brussels: Workflow Management Coalition, 2004. 103-109.

[2] Kim Yogh, Kang Suk-Ho, Kim Dongsoo. WW-FLOW: Web Based Workflow Management with Runtime Encapsulation [J]. Internet Computing IEEE, 2000, 4(3): 55-64.

[3] Tagg R, Lelatanavit W. Using an Active DBMS to Implement a Workflow Engine [C]. In: Database Engineering and Applications Symposium. 1998 Proceedings. New York : IDEAS International, 1998. 286-295.

[4] Robert Muller, Ulrike Greiner, Erhard Rahm. A Workflow System Supporting Rule-based Workflow Adaptation [J]. Data & Knowledge Engineering, 2004, 51(2): 223-256.

[5] FAN Yu-shun, CHENG Wu. Research on a Workflow Modeling Method to Improve System Flexibility [J]. Journal of Software, 2002, 13(4): 833-839.

A workflow management system based on message notification

LIU Li-zhi, LIU Jun

(School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan 430074, China)

Abstract: This paper introduces a workflow management system based on message notification. Message notification mechanism is not only the core of the system but also drives business workflows, thus we design workflow module definition and workflow engine according to it. If we use task to denote a form instance processed in a flow node, the process of the form instance in the flow node can be expressed by task state. Workflow engine scans all tasks under process in a fix period to control the form processed according to workflow module.

Key words: message notification mechanism; workflow module definition; workflow engine

本文编辑：陈晓苹