一种基于MM&MBPNN的软件衰退预测方法研究

林已杰[1,2,3] 张为群[1,3] 周 敏[1,2,3] 赖 清[4]

(西南大学计算机与信息科学学院 重庆 400715)[1] (西南大学信息中心 重庆 400715)[2]

(重庆市智能软件与软件工程重点实验室 重庆 400715)[3] (重庆医科大学计算机教研室 重庆 400016)[4]

摘 要 由于系统环境的恶化,运行中的软件系统不可避免地会出现衰退现象。针对该现象,提出了一种用基于放大误差信号的改进的BP神经网络(MBPNN)来表示软件状态的马尔科夫模型(MM),并通过此方法来预测软件衰退。此方法弥补了单纯使用马尔科夫方法时对软件衰退状态预测不够准确的缺点,为软件抗衰的实施提供了依据。仿真实验表明,这是一种有效可行的预测方法。

关键词 软件衰退,软件抗衰,马尔科夫模型,BP神经网络

Study on Software Aging Forecasting in Markov-Model-based MBP Neural Network Methods

LIN Yi-jie[1,2,3] ZHANG Wei-qun[1,3] ZHOU Min[1,2,3] LAI Qing[4]

(College of Computer and Information Science, Southwest University, Chongqing 400715, China)[1]

(Information Center, Southwest University, Chongqing 400715, China)[2]

(Chongqing Intelligent Software and Software Engineering Laboratory, Chongqing 400715, China)[3]

(Chongqing Medial University, Chongqing 400016, China)[4]

Abstract Becouse of the degeneration of system environment, running software system doesn't avoid software aging. This paper gave a reforming MBP Neutral Network(MBPNN) to represent the Markov Model(MM). This method can be used to supply the gap of the inaccuracy of forecasting result, when only using Markov model. According as the result, software rejuvenation can be implemented reasonably. Study results indicate that the method can forecast the status of running software system efficiently.

Keywords Software aging, Software rejuvenation, Markov model, BP neural network

软件衰退是指软件系统在长期不间断的运行过程中,由于操作系统内存的不紧凑占用和泄漏、未释放的文件锁、数据更新不及时、存储空间碎片以及舍入误差的累积等原因造成软件系统运行环境"恶化",从而导致运行中的软件系统性能衰退[1,2]。例如持续提供服务的Web服务器,在系统正常运行一段时间后,由于软件系统衰退,出现服务质量下降、应答时间增加或拒绝服务等现象,结果必然导致高成本的停机重启。软件衰退对整个系统的可靠性构成很大威胁,绝对地避免运行中的软件系统衰退是不可能的。为了对抗软件衰退,减少由其导致的系统意外终止所带来的损失,提出了软件抗衰的思想[3]。

本文设计了一种马尔科夫和BP神经网络相结合的软件系统衰退预测方法,建立了一个改进的BP神经网络并对马尔科夫模型进行表示,优化权值调节以得到期望值。仿真实验表明,通过本文方法得到的软件系统(以下简称系统)衰退预测结果更准确,为软件抗衰的实施提供了依据,是一种有效可行的软件衰退预测方法。

1 软件衰退相关概念描述

要预测运行中的软件系统衰退,首要是对软件衰退的概念进行描述。软件是否处于衰退状态,是通过目前的系统状态来进行判定的,所以这里先给出有关系统性能阈值[4]、系统状态、系统状态划分、系统状态概率阈值的定义。

定义1(系统性能阈值) 若系统在给定的系统资源集G下处理对象$C(t)$的能力极限为C_{max},则对于任意时刻t系统的性能状态定义为:$L(t)=C(t)/C_{max}$,其中资源集G是资源g_i的集合,即$G=\{g_1,g_2,\cdots,g_n\}$。若系统处理对象$C(t)$消耗的资源$G(t)=\{g_1(t),g_2(t),\cdots g_m(t)\}$,$m\leqslant n$,则系统的性能状态可以表示为:

$$L_t=f(.)\left(\frac{g_1(t)}{g_1},\frac{g_2(t)}{g_2},\cdots,\frac{g_m(t)}{g_m}\right)$$

其中,$f(.)$表示资源元素的消耗量与性能状态之间的映射关系;g_i表示资源元素i的最大配置限度。给定系统性能状态对应的L_i,称之为系统处于状态i的性能阈值。

定义2(系统状态) 根据定义1,给定系统性能阈值集合$(L_0,L_1,\cdots,L_n)$,且$L_0<L_1<\cdots,L_n$,$L_0=0$,$L_n=1$。对于任意时刻t有$L(t)$,$\forall L(t)\in[L_m,L_{m+1}]$,$m\in[0,n-1]$,则称系统在$t$时刻的状态$S$为$S_m$。

定义3(系统状态划分) 根据定义1和定义2,给出软件系统的性能阈值集合(L_0,L_1,L_2,L_3),且$L_0<L_1<L_2<L_3$,

到稿日期:2009-07-21 返修日期:2009-09-01 本文受重庆市自然科学基金项目(CSTC,2006BA2003)资助。

林已杰(1980-),男,硕士生,主要研究方向为软件理论等,E-mail:laoniupi@gmail.com;张为群(1950-),男,教授,主要研究方向为软件理论、软件测试等;周 敏(1980-),男,硕士生,主要研究方向为软件理论等。

$L_0=0, L_3=1$，系统状态集合(S_0, S_1, S_2)分别为：

$S_0: L_{S0}\in[L_0, L_1]$，系统处于健康态发生衰退的概率基本为0。

$S_1: L_{S1}\in(L_1, L_2]$，系统处于亚健康态，经过一段较长时间的运行，软件系统开始衰退。

$S_2: L_{S2}\in(L_2, L_3]$，系统处于衰退态，系统性能严重降低。

定义4(系统状态概率阈值) 根据定义3，t时刻系统状态为S，$S\in(S_0, S_1, S_2)$，$p_i(t)$为系统处于状态S_i的概率，则有：

$$p_i(t)=P(S_i, 0\leqslant i\leqslant 2), \sum_{i=0}^{2} p_i(t)=1$$

称$(p_0(t), p_1(t), p_2(t))$为t时刻系统的状态概率集合，设定数值$\mu(0\leqslant\mu\leqslant 1)$为系统状态概率阈值。

推理1(软件衰退状态判定) 根据定义4，给出系统的状态概率集合$(p_0(t), p_1(t), p_2(t))$和μ，$\forall(p_0(t)+p_1(t)\leqslant\mu)\Rightarrow S=S_2$，此时系统处于衰退状态。

2 基于放大误差信号的BP神经网络改进(MBPNN)算法

BP(Back Propagation)神经网络是一种按误差逆向传播算法训练的多层前馈网络，是目前应用最广泛的神经网络模型之一[6]。BP网络能学习和记忆大量的输入-输出模式映射关系，无需事前揭示描述这种映射关系的数学模型。它的学习规则是使用梯度下降法，通过误差信号的反向传播来不断调整网络的权值和阈值，使网络的误差平方和最小，从而使网络权值收敛到最优状态。但在实践过程中发现，BP算法存在两方面的问题：首先，它不能确保神经网络能够收敛到误差函数的全局极小值。其次，由于BP神经网络采用最速梯度下降法，此算法在目标函数搜索时使用的是固定参数法，因此不能保证整体误差函数具有单调下降的特性，这将影响到用于软件系统状态预测的神经网络的精度与效率。针对以上缺陷，本文提出了一种基于放大误差信号的BP神经网络改进算法。

2.1 放大误差信号方法的加入

所谓加入放大误差信号项是通过给原信号处理函数δ_{pk}，δ_{pj}的导数值乘上一个大于1的放大因子，以此来增大权值的调节幅度，即：

原误差信号函数：

$$\begin{aligned}\delta_{pk}&=y_{pk}(1-y_{pk})(t_{pk}-y_{pk})\\ \delta_{pj}&=h_{pj}(1-h_{pj})\sum_{k=1}^{M}\delta_{pk}\omega_{jk}\end{aligned}\tag{1}$$

改进后的误差信号函数：

$$\begin{aligned}\delta_{pk}'&=-\ln(y_{pk}(1-y_{pk}))\cdot y_{pk}(1-y_{pk})(t_{pk}-y_{pk})\\ \delta_{pj}'&=-\ln(h_{pj}(1-h_{pj}))\cdot h_{pj}(1-h_{pj})\sum_{k=1}^{M}\delta_{pk}\omega_{jk}\end{aligned}\tag{2}$$

其中，δ_{pk}'，δ_{pj}'为改进后的误差信号，h_{pj}表示隐含层第j个节点的输出，t_{pk}和y_{pk}分别代表输出层第k个节点的目标输出以及实际输出，ω_{jk}表示隐含层第j个节点与输出层第k个节点之间的连接权值。可以看出，这里将原误差信号δ_{pk}，δ_{pj}中的因子$f(1-f)$放大了$-\ln(f(1-f))$倍。注意到当$f\to 0$时，放大因子趋于无穷，这说明当神经元的输出为0或者1时，误差信号放大的倍数迅速增大，误差信号将会被放大到对权值修正有明显影响的范围，这样就可以在一定程度上加快算法的收敛速度。

2.2 改进算法的收敛性分析

我们将通过定理1证明本文所提出的改进BP网络算法较标准BP网络算法具有更快的收敛速度，能更好地应用于软件衰退的预测。

定理1 改进的BP神经网络算法(MBP)与标准BP网络算法(BP)相比，以下关系成立：

$$\left|\left(\frac{\partial E}{\partial t}\right)_{MBP}\right|>\left|\left(\frac{\partial E}{\partial t}\right)_{BP}\right|$$

证明：根据改进BP算法的权值更新公式，输出层误差信号在更新前后具有以下关系：

$$(\delta_{pk})_{MBP}=-\ln(y_{pk}(1-y_{pk}))\cdot(\delta_{pk})_{BP}$$

考虑某个确定的样本p，则误差对隐含层到输出层权值的一阶导数为：

$$\begin{aligned}\left(\frac{\partial E_p}{\partial\omega_{jk}}\right)_{MBP}&=h_{pj}(\delta_{pk})_{MBP}=-h_{pj}\ln(y_{pk}(1-y_{pk}))(\delta_{pk})_{BP}\\ &=-\ln(y_{pk}(1-y_{pk}))\left(\frac{\partial E_p}{\partial\omega_{jk}}\right)_{BP}\end{aligned}$$

同理可证误差对于输入层到隐含层权值的一阶导数为：

$$\left(\frac{\partial E_p}{\partial\omega_{ij}}\right)_{MBP}=-\ln(h_{pj}(1-h_{pj}))\left(\frac{\partial E_p}{\partial\omega_{ij}}\right)_{BP}$$

将误差对所有可调权值的导数写成矩阵形式：

$$\left(\frac{\partial E_p}{\partial\omega}\right)_{MBP}=K\left(\frac{\partial E_p}{\partial\omega}\right)_{BP}$$

其中，K为一个对角元素大于1、其余元素为0的对角矩阵。又因为误差函数关于变量t的一阶导数公式为：

$$\frac{\partial E}{\partial t}=\left(\frac{\partial E}{\partial\omega}\right)^T\frac{\partial\omega}{\partial t}=\left(\frac{\partial E}{\partial\omega}\right)^T\left(-\eta\frac{\partial E}{\partial\omega}\right)$$

所以有：

$$\left|\left(\frac{\partial E}{\partial t}\right)_{MBP}\right|-\left|\left(\frac{\partial E}{\partial t}\right)_{BP}\right|=\eta\left(\frac{\partial\omega}{\partial t}\right)^T(K^TK-I)\frac{\partial E}{\partial\omega}>0$$

从而可以得出结论：MBP的误差变化率大于标准BP算法，具有更快的收敛速度。

3 马尔科夫预测模型的MBPNN表示

3.1 马尔科夫预测模型

马尔科夫链预测是根据初始的状态概率向量和状态概率转移矩阵来推测某一变量未来某一时期所处状态的一种方法，其理论基础是马尔科夫过程，它描述的是一个随机时间序列的动态变化过程。

文献[3]根据运行中的系统状态建立了基于连续时间的马尔科夫链模型。该模型采用了连续时间的马尔科夫链来描述系统运行过程中的状态关系。系统运行过程可划分为3个阶段[5]。各个状态之间的转移模型如图1所示。

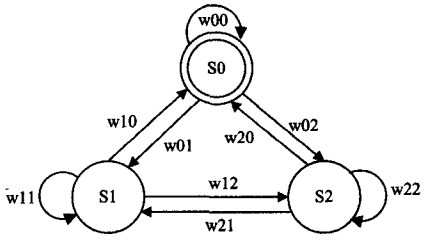

图1 软件运行时的状态转移模型

上面的马尔科夫模型的状态转移方程可以表示为式(3)，它表示了系统状态变化的动态过程：

$$\begin{cases} p_0(t+\Delta t)=w_{00}p_0(t)+w_{01}p_1(t)+w_{02}p_2(t) \\ p_1(t+\Delta t)=w_{10}p_0(t)+w_{11}p_1(t)+w_{12}p_2(t) \\ p_2(t+\Delta t)=w_{20}p_0(t)+w_{21}p_1(t)+w_{22}p_2(t) \end{cases} \quad (3)$$

其中，$p_i(t+\Delta t)$ $(0\leqslant i\leqslant 2)$ 是系统在 $t+\Delta t$ 时刻处于状态 S_i 的概率，当 $t=0$ 时，$p_0=1$，$p_1=0$，$p_2=0$。w_{ij} 描述了从状态 i 到状态 j $(0\leqslant i,j\leqslant 2)$ 的转化概率。

基于以上系统的马尔科夫模型，假设系统的初始状态用向量 S_S 来描述，系统运行 T 时间长度后，它所期望的状态可以用 $S_e=(a_0,a_1,a_2)$ 来描述，那么系统状态可以定义为求解矩阵

$$W=\begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix}$$

中的 w_{ij} 值，使其满足：

$$S_sW^k=S_e \quad (4)$$

其中，$k=T/\Delta t$ 。

3.2 MBPNN 的表示模型

我们往往需要借助于一定数学方法通过化简来求解式(4)。但是简化的马尔科夫模型通常会失去实际系统本身的一些动态特性，从而求解不会得到十分精确的结果[7]。

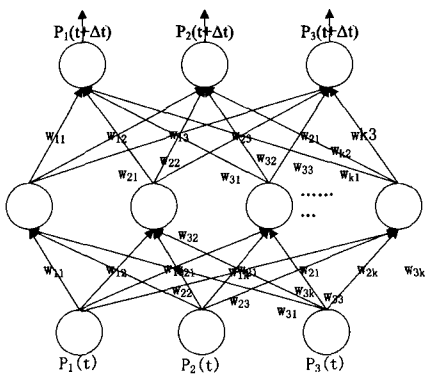

图 2　马尔科夫模型的神经网络表示

为了克服上述困难和问题，本文提出用图 2 所示的神经网络模型来表示图 1 所示的马尔科夫模型，每层均有 n 个神经元分别与马尔科夫模型中的 n 个状态相对应。权值 w_{jk} 与图 1 中状态 S_n 之间的状态转换概率 P_{ij} 相对应。假设该神经网络模型在 t 时刻的输入向量为 $X(t)$，在 $t+\Delta t$ 输出向量为 $Y(t+\Delta t)$，从图 2 可以看出它们和马尔科夫模型在 t 时刻以及 $t+\Delta t$ 时刻的状态有如下对应关系：

$$\begin{cases} X(t)=(P_1(t)P_2(t)P_3(t)\cdots P_n(t)) \\ Y(t+\Delta t)=(P_1(t+\Delta t)P_2(t+\Delta t)P_3(t+\Delta t)\cdots P_n(t+\Delta t)) \end{cases} \quad (5)$$

输入层神经元 i 的处理特性为：

$$O_i=x_i \quad (6)$$

输出层神经元 j 的处理特性为：

$$y_j=\sum_{i=1}^{n}w_{ji}o_i \quad (7)$$

其中，x_i 和 y_j 分别是输入向量 $X(t)$ 和输出向量 $Y(t+\Delta t)$ 中的元素。

结合式(5)－式(7)，可将上述基于马尔科夫模型的神经网络计算过程描述为：

$$\begin{cases} Y(t+\Delta t)=X(t)W \\ X(t+\Delta t)=Y(t+\Delta t) \end{cases} \quad (8)$$

不难看出，对于上述神经网络模型而言，其目标是寻求合适的 W，使得上述计算过程迭代 n 次后有：

$$Y(T)=S_e \quad (9)$$

为寻求合适的 W，必须重复利用神经网络具有的学习功能和自适应功能，对神经网络中的连接权值进行适当调整。如果能够通过权值调整使得 $E=0$，那么上述神经网络模型中的各个连接权值必须满足式(9)，这些连接权值也就是系统状态转移概率所要求的解。

综上所述，基于马尔科夫 BP 神经网络模型的软件状态预测算法流程图可以描述为图 3。

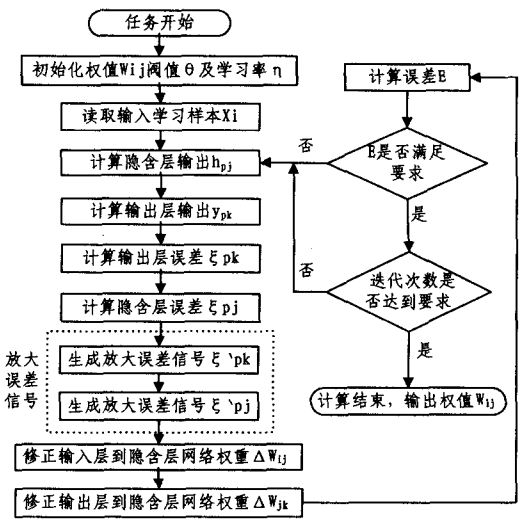

图 3　用于软件状态预测的 MM&MBPNN 算法流程图

4　仿真实验

为验证上述方法的有效性，我们通过仿真实验来预测软件系统的状态。假设以 1h 为时间间隔，观察 50 个周期，得到系统状态的转移概率矩阵的集合为：

$$W=(W_1,W_2,\cdots,W_{50})$$

系统状态概率的集合为：

$$S=(S_1,S_2,\cdots,S_{50})$$

其中，$S_i=(p_{i1},p_{i2},p_{i3})$，$1\leqslant i\leqslant 50$。

若已知时刻 45 时的系统状态概率为 $S=(0.52 \quad 0.32 \quad 0.16)$，转移概率矩阵为：

$$W_\lambda=\begin{bmatrix} 0.75 & 0.2 & 0.05 \\ 0.07 & 0.68 & 0.25 \\ 0.02 & 0.05 & 0.93 \end{bmatrix}$$

根据本文第 1 节马尔科夫模型可以预测出后 5 个周期时的系统状态分别为：

$$S_{46}=SW_\lambda;S_{47}=SW_\lambda^2;S_{48}=SW_\lambda^3;S_{49}=SW_\lambda^4;S_{50}=SW_\lambda^5$$

根据本文第 2 节的 BP 神经网络模型选取前 45 个数据来训练神经网络，然后模拟后 5 个数据。利用 MATLAB 工具进行计算。所采用的神经网络为 3 层网络，输入层包含 3 个神经元，输出层包含 3 个神经元。为了研究不同的隐含层神经元数量与最终性能的关系，以确定最优隐含层规模，我们分别采用隐含层规模为 3,5,7,10 的神经网络进行了对比试验。此外，神经网络的初始权值根据在工程领域常用的 BP 网络随机化初始方法来确定。当在两次迭代之间网络输出没有变化时，停止 BP 神经网络的学习。

针对所采用的数据集，使用传统马尔科夫方法和具有不

同隐含层规模的 BP 神经网络算法分别进行了 130 次独立实验，并对实验结果进行了统计分析，结果如图 4 所示。

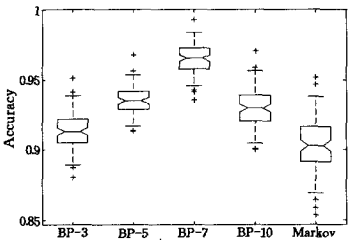

图 4　基于测试数据的仿真试验结果

从图 4 中可以看出，不论采用何种隐含层规模的 BP 神经网络，其最终预测精度均高于单纯的马尔科夫方法，其中 4 种神经网络所达到的平均预测精度分别为 91.4%，93.5%，96.3%以及 92.9%，而马尔科夫方法的平均预测精度为 90.3%。以上结果充分证明了神经网络方法在软件抗衰中的性能优势。

此外，我们可以看到在 4 种不同隐含层规模的神经网络中，规模为 7 的神经网络达到了最优性能。这是因为如果隐含层神经元过少，网络的学习性能不足，从而网络权值将无法达到最优；而如果隐含层神经元过多，网络的学习性能将达到"过饱和"状态，这将导致网络过早地陷入局部最优状态。以上的实验结果为我们选取最优的隐含层规模提供了参考依据。

结束语　从本文可以看出，运行中的软件系统状态的马尔科夫模型可由 MBP 神经网络来表示并求解。利用本文提出的 MBP 神经网络对马尔科夫预测模型进行不断训练以达到期望值，利用此方法进行软件系统衰退预测，可以克服单纯使用马尔科夫模型时预测不够准确的问题，解决以往方法所面临的一些困难，它是一种值得深入研究的软件系统衰退预测方法。

参 考 文 献

[1] Avritzer A, Weyuker E J. Monitoring smoothly degrading systems for increased dependability [J]. Empirical Software Eng, 1997,12(1):55-77

[2] Garg S, Puliafito A, Telek M, et al. Analysis of preventive maintenance[J]. IEEE Trans on Computers, 1998,47(1):96-107

[3] Huang Y, Kintala C, Kolettis N, et al. Software rejuvenation: analysis module and applications[C]// IEEE Intl. Symposium on Fault Tolerant Computing. 1995:381-390

[4] 王田. SMSC 负荷状态检测方法研究[J]. 电子科技大学学报, 2003,32(2)

[5] 徐建，张坤，刘玉凤. 软件抗衰研究综述[J]. 小型微型计算机系统，2007,28(11)

[6] 阎平凡，张长水. 人工神经网络与模拟进化计算[M]. 北京：清华大学出版社，2005:26-30

[7] 戴葵，仇广煜，胡守仁. 一种基于离散马尔科夫模型的神经网络可靠性设计方法[J]. 计算机工程与科学，1999,21(3)

[8] 周成容. BP 神经网络的模糊改进及应用[J]. 重庆工学院学报：自然科学版，2008,22(6):153-156,158

(上接第 142 页)

选择性恢复功能根据事务间的依赖关系来决定需要撤销的事务，以便最大限度地保留无辜事务所作的工作，这对传统日志机制提出了新的需求。现有自修复数据库原型系统的实现方法在性能及通用性等方面存在着各种问题，本文在分析了现有方法不足的基础上，提出了一种新的日志机制，该机制中增加了事务依赖日志，并以前像表日志代替传统的回滚段，使系统能以统一的方式兼顾传统恢复功能和选择性恢复功能。性能分析表明，本方法在时间性能和空间性能上都优于现有的 ITDB 方法。与基于 MVCC 机制的 Phoenix 方法相比，本方法适用于主流的基于封锁机制的 DBMS，因而具有更好的通用性。

参 考 文 献

[1] Liu P, Jing J. Architectures for Self-healing Databases under Cyber Attacks[J]. International Journal of Computer Science and Network Security, 2006,6(1B):204-216

[2] Smirnov A, Chiueh T. A Portable Implementation Framework for Intrusion-Resilient Database Management Systems[C]// Proc. of the 2004 International Conference on Dependable Systems and Networks (DSN'04). 2004:443-452

[3] Bernstein P A, Hadzilacos V, Goodman N. Concurrency Control and Recovery in Database Systems[M]. Reading, MA: Addison-Wesley, 1987

[4] Luenam P, Liu P. ODAM: An On-the-fly Damage Assessment and Repair system for Commercial Database Applications[C]// Proc. 15th IFIP WG 11.3 Working Conference on Database and Application Security. July 2003

[5] Amman P, Jajodia S, Liu P. Recovery from Malicious Transactions[J]. IEEE Transactions on Knowledge and Data Engineering, 2002,14(5):1167-1185

[6] Chiueh T, Pilania D. Design, Implementation, and Evaluation of A Repairable Database Management System[C]// Proc. of the 21st International Conference on Data Engineering (ACSAC 04). 2005:1024-1035

[7] Chiueh T, Bajpai S. Accurate and Efficient Inter-transaction Dependency Tracking[C]// Proc. of the 2008 IEEE 24th International Conference on Data Engineering (ICDE 08). 2008:1209-1218

[8] Wang H, Liu P, Li L. Evaluating the survivability of Intrusion Tolerant Database systems and the impact of intrusion detection deficiencies[J]. International Journal of Information and Computer Security, 2007,1(3):315-340

[9] Liu P, Amman P, Jajodia S. Rewriting Histories: Recovering from Malicious Transactions[J]. Distributed and Parallel Databases, 2000,8(1):7-40

[10] Oracle9i LogMiner[EB/OL]. http://otn.oracle.com/products/oracle9i/daily/oct25.html