第6卷第10期
1995年10月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 6, No. 10
Oct. 1995

关于实时数据库事务*

刘云生　　K. Ramamritham　　J. Stankovic

（华中理工大学计算机系，武汉 430074）　　（Massachusetts 大学计算机科学系，Amherst，美国）

摘要　实时数据库事务显示了与传统数据库事务的很大不同. 这些不同主要表现在事务的复杂结构、定时特性、相关性和正确性方面. 本文先分析了实时数据库事务的应用特征与需求，并给出了一个复杂事务结构的框架，然后着重讨论实时事务间的相关性：结构相关、数据相关、行为相关，以及实时事务的结果、结构、行为及时间正确性.

关键词　数据库，实时数据库，事务，实时事务，复杂事务，事务正确性.

实时数据库事务组合了实时任务和传统数据库事务两者的特征，但并非它们在概念、机制、技术上的简单集合，而有一系列的问题需要研究与解决，如扩展事务模型及其结构特征，事务处理的硬软限制、事务间的关系、事务正确性和数据一致性及其与数据语义和事务特征的联系，事务调度的协议和算法. 这些问题彼此且还与其它方面的问题紧密相关. 因此，实时数据库事务处理比传统的实时任务和数据库事务都更复杂、更困难. 它们必须同时实现数据一致性，包括外部、相互和动态一致性[1]，和定时限制两者. 近年来，许多研究者已对实时事务处理尤其是调度与并发控制作了大量的研究. 本文侧重讨论实时事务的概念与含义、结构、特性和正确性等方面.

1　实时数据库事务的应用分析

实时数据库事务的特征及其处理与实时应用语义紧密相关，故我们必须首先进行应用分析以明确其性质与要求，从而确定其可能的模型结构、特性及处理要求. 我们从下列方面来分析实时应用活动（事务）：

(1)结构　实时任务往往具有结构上的相互联系，无结构的、原子的和隔离的传统事务模型不完全适用，而有时要求嵌套或层次结构事务，有时要求分裂和合并、通信与数据交换、或长寿事务等. 因此必须开发各种复杂事务模型.

(2)计划安排　实时活动（事务）具有各种要求与限制，如计算时间要求、资源要求、通信要求、执行次序限制、执行时间限制、定位限制（在分布式系统情况下）等. 我们必须事先知道

* 本文 1994－03－10 收到，1994－07－21 定稿

本研究是国家自然科学基金与国防预研资助项目. 作者刘云生，1940年生，教授，主要研究领域为数据库与信息系统，高级应用的数据库（主动、实时、时序）技术. K. Ramamritham，1955年生，教授，主要研究领域为数据库与实时系统，实时数据库事务处理. J. Stankovic，1951年生，教授，主要研究领域为分布式与实时计算，实时数据库.

本文通讯联系人：刘云生，武汉 430074，华中理工大学计算机系

© 中国科学院软件研究所 http://www.jos.org.cn

事务何时被执行、有何种要求与限制,以便能计划安排各种事务.

(3)分布率　实时任务(事务)通常是周期的,但也有非周期或随机的,有的是长寿的,为了调度必须事先知道各种事务的类型及事务到达的分布率.对于长寿事务,这无关紧要.对于周期事务,问题也好解决,但对非周期和随机事务则很困难.可以将一个非周期事务的两个体现(具体执行)间的最小间隔时间视作其周期,从而将所有的硬实时事务都作为周期事务来调度.

(4)定时特性　实时事务的实时特性有:事务的松缓度、截止期的粒度和严格度(即硬软性)、优先顺序限制、终点到终点的定时限制、价值函数的形状等,这些是定义和处理实时事务(甚至实现实时数据库系统)的基本因素.

(5)可预报性　硬实时事务的定时限制必须确保,因而就要预测这种事务是否会满足其截止期.这要求事先知道事务的最坏情况执行时间及所需数据与资源,并要求这种最坏情况预测与实际的差别尽可能小.对软实时事务,预测也是重要的,它可使满足其截止期的软实时事务数尽可能大.虽然这种预报具有静态可确定性,但要动态确定是很难的.因为数据库系统中有许多不可动态预测的因素[2],必须弄清产生不可预报的因素,从而采取有效措施来克服它们.

(6)合作性　实时事务总是"合作"进行的,除了前面所述的结构联系外,彼此间还有许多联系,如共享数据联系、通信联系、时间上的联系等.必须弄清事务间存在何种联系.

(7)不可逆性　实时应用中的许多活动是不可逆的.例如对应于火箭"点火"的事务、记录飞行体的位置、速度、方向的事务等都是不可逆的.对于这种事务,还原/重启动是毫无意义的,必须为实时数据库事务恢复开发新的概念、技术和方法.

基于以上分析,可以得出实时应用环境下,数据库事务有下列要求或性质:

(1)有复杂的结构(模型),如层次和嵌套,分裂/合并等;

(2)必须彼此协调、合作地并发执行,故要求强的可见性[2];

(3)及时性比正确性更重要,有时宁愿及时地获得部分正确的信息,而不愿要正确但已过时(无效)的信息;

(4)因上述各点,事务调度和并发控制的可串行化协议不一定必要,而必须要"识时"协议和"时间正确"的调度与并发控制;

(5)一般地数据是二维的:值与时间.除值以外,每个数据还有一个"及时"的时区.大多数情况下事务只允许存取"及时"数据,故事务必须能处理数据的时标(time－stamp)和时区(interval);

(6)事务恢复不一定就意味着数据库状态的完全复原;

(7)需要预分析和预计划事务以便所需要的数据、计算时间及处理资源在适当的时候是可用的,因而能按截止期完成.

2　复杂事务结构

复杂的实时应用常常包含一些反复的、开端的(open－ended)和彼此合作的活动,有的还包含了"子活动",且子活动又依次地包含子活动,即它们形成层次结构.因而,具有 ACID

© 中国科学院软件研究所 http://www.jos.org.cn

特征[2]的传统事务模型对复杂的实时应用已不适合，为此对传统模型已进行了各种扩展[3-6]. 与传统的事务是原子的、平淡的数据库操作序列的定义相反，复杂事务定义为可以形成各种"内部结构"的数据库操作集合. 即复杂事务中的操作可按应用语义而成组，且这个组又可以是一个复杂事务(称为子事务)，组之间可有各种联系. 因此复杂事务可能构成一个层次的复杂网状结构，传统事务仅是其最简单的特例.

为了提供一个复杂事务结构的统一框架和该框架的一个形式说明，我们引入"循环"和"开端"事务概念：

一个"循环"事务(looped transaction)就是一个循环地执行的一般或结构事务. 它象一个周期事务，但一个执行的结束是下次执行的开始，且是一个整体.

一个"开端"事务(open－ended transaction)就是一个具有很长执行期的一般事务或结构事务. 循环事务和开端事务统称"长寿"事务，我们可以形式地表示复杂事务结构如下：

TRANSACTION::=〈COMP－TRAN〉

COMP－TRAN::=〈GENERAL－TRAN〉|〈LONGLIFE－TRAN〉|〈STRU－TRAN〉

GENERAL－TRAN::=〈O,S,D,C〉

其中O,S,D和C定义为：O：一个操作集，包括数据库操作和事务管理操作. S：O上的一个偏序. D：由O中操作所存取的数据集. C：由该事务维持的限制集，包括数据完整性/一致性限制和定时限制等.

LONGLIFE－TRAN::=〈LOGICAL－L－TRAN〉|〈PHYSICAL－L－TRAN〉

LOGICAL－L－TRAN::=Looped Transaction

PHYSICAL－L－TRAN::=Open－Ended Transaction

STRU－TRAN::=〈MULTILEVEL－TRAN〉|〈NESTED－TRAN〉|〈SPLIT－TRAN〉|〈JOINT－TRAN〉|〈COMMU－TRAN〉|〈COOP－TRAN〉

MULTILEVEL－TRAN::=({operation,},{〈MULTILEVEL－TRAN〉,})

NESTED－TRAN::=({〈GENERAL－TRAN〉,},{〈NESTED－TRAN〉,})

SPLIT－TRAN::=〈SERIAL－S－TRAN〉|〈INDEP－S－TRAN〉

SERIAL－S－TRAN::=〈GENERAL－TRAN〉→→〈GENERAL－TRAN〉〈GENERAL－TRAN〉

INDEP－S－TRAN::=〈GENERAL－TRAN〉→→(〈GENERAL－TRAN〉,〈GENERAL－TRAN〉)

JOINT－TRAN::=(〈GENERAL－TRAN〉,〈GENERAL－TRAN〉)→→〈GENERAL－TRAN〉

COMMU－TRAN::=(〈SENDER－TRAN〉→〈RECEIVER－TRAN〉)

SENDER－TRAN::=〈GENERAL－TRAN〉

RECEIVER－TRAN::=〈GENERAL－TRAN〉

COOP－TRAN::={(〈COMPONENT－TRAN〉←→{〈SUB－COOP－TRAN〉,}),}

COMPONENT－TRAN::=〈GENERAL－TRAN〉

SUB－COOP－TRAN::=〈COMPONENT－TRAN〉|〈COOP－TRAN〉

这里"→→"表示"IS－TRANSFORMED－INTO"关系，"←→"和"→"分别表示"EXCHANGES－DATA－WITH"和"SENDS－DATA－TO"关系. 应当指出的是：

(1)尽管嵌套事务和多层事务都有层次结构，但它们是有区别的：(a)前者的层次结构是"显式"的，是一种用户可用的技术，而后者的层次内部结构是"隐式"的，作为一种系统设施而提供；(b)前者的部件是事务，不必是原子的，而后者的部件是操作，这些操作又可分解为"子操作"，但不管操作还是子操作都认为是原子的；(c)前者的结构是为了使失败局部化和利用事务的并行机制，而后者的结构是为了实现复杂对象上的操作.

© 中国科学院软件研究所 http://www.jos.org.cn

(2)合作事务也有类似于嵌套事务那样的层次结构,但它没有如嵌套事务那样的对象继承性(子事务的可存取对象集包括其祖先已存取了的所有对象),且合作事务中数据交换的"数据流"也只有在相邻层的部件事务间存在.

(3)两个分裂事务之一是原事务.合并事务是两个原事务之一.

3　事务的定时限制

定时性是实时数据库的基本特征,与事务相连的定时性表现在两个方面:

事务定时限制:由需要不断跟踪外部环境所施加于系统的反应时间要求而引起.这种定时限制典型地取施加于非周期事务的截止期形式.截止期是一个时区,它看作是对"将来"的一种限制,如"飞机必须在10秒内完成其着陆准备".

数据的时间一致性:由于要保持由数据库的内容所反映的状态与外部环境的实际状态一致而引起.它可看作是对"过去"的一种限制的一个时间窗口.这类定时限制常常以周期的形式给出,如"每5秒取样速度1次".定时限制要求系统具备时间处理机制和"识时"的事务处理.不同的事务表现出不同的定时限制特性:硬、软、固[7],这取决于事务超截止期所产生的结果.

硬截止:事务超截止期将给系统带来灾难性后果,或说其价值函数取负值.它对应于安全危急性活动.

软截止:事务超截止期后仍具一定的价值,直至某个时刻价值降为零,且此后一直保持为零而不为负.

固截止:一旦事务到达其截止期,则对系统失去意义,其价值固定为零.实质上它是软截止的一种特例.

图1描绘事务截止期的硬、软、固特性.

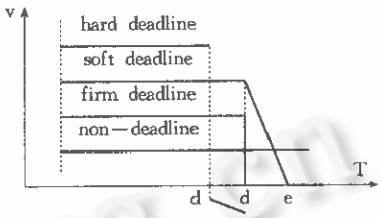

图1　事务截止期类型

要使所有的硬、软截止期事务都满足其截止期是很困难的,系统的设计目标应该是确保所有硬截止期,而使不满足其截止期的软截止期事务的个数尽可能小.

4　事务的相关性

实时数据库事务之间有着各种关系,包括结构关系、数据与通信关系、时间关系等,这些关系带来了各种事务间的相关性.

4.1　结构相关

它由事务模型的结构特征而引起,且提供了一个说明和解释复杂事务模型下并发事务行为的一种方便的方法.不同的复杂事务模型有不同的结构相关性,但它们可以通过一些基本的事务依赖性来说明.这些基本事务依赖性是:

(1)开始依赖:事务 t_1 开始意味着 t_2 先开始,记为 $t_1\ BD\ t_2$;

(2)提交依赖:只有事务 t_2 提交或夭折后 t_1 才能提交,记为 $t_1\ CD\ t_2$;

(3)夭折依赖:事务 t_2 夭折包含了 t_1 的夭折,记为 $t_1\ AD\ t_2$.

以上3个是最基本的事务依赖,下面再给出几个有用的基本依赖.

© 中国科学院软件研究所 http://www.jos.org.cn

(4)排他夭折依赖:若 $t_1\ AD\ t_2$ 且对于任何 $t'_2 \neq t_2$,有 $t_1\ \not{AD}\ t'_2$,则说 t_1 对 t_2 排他夭折依赖,记为 $t_1\ AD^x\ t_2$;

(5)传递夭折依赖:若 $t_1\ AD\ t_2$ 且 $t_2\ AD\ t_3$,则显然 t_1 传递地夭折依赖于 t_3,称为 t_1 对 t_3 有传递夭折依赖,记为 $t_1\ AD^*\ t_3$;

(6)传递提交依赖:若 $t_1\ CD\ t_2$ 且 $t_2\ CD\ t_3$,则类似于传递夭折依赖,t_1 对 t_3 有传递提交依赖,记为 $t_1\ CD^*\ t_3$.

下面分别讨论第2节所述各种复杂事务结构的结构相关性:

(1)嵌套事务的结构相关 设 tp 是一个父事务,T_c 是 t_p 的子事务集,则嵌套事务的结构相关可描述为:$\forall\ t_c \in T_c (t_c\ BD\ t_p) \wedge (t_c\ AD^x\ t_p) \wedge (t_p\ CD\ t_c)$

(2)分裂事务的结构相关 设事务 t 分裂为 t_b 和 t_c,则对

(a)独立分裂:$(t_b\ BD\ t) \wedge (t_c\ BD\ t)$

(b)顺序分裂(设 t_b 先于 t_c):$(t_b\ BD\ t) \wedge (t_c\ BD\ t) \wedge (t_c\ CD\ t_b) \wedge (t_c\ AD^*\ t_b)$

(3)合并事务的结构相关 设事务 t_a 和 t_b 合并成 t,则$(t\ BD\ t_a) \wedge (t\ D\ t_b) \wedge (t_a\ AD^*\ t) \wedge (t_b\ AD^x\ t)$

(4)通信事务的结构依赖 设 t_r 和 t_p 分别为接收和发送事务,则$(t_r\ AD\ t_s) \wedge (t_s\ CD\ t_r)$

(5)合作事务的结构依赖 设 t_i 为一高层事务,t_{ij} 是 t_i 的直接低层部件事务,则$\forall\ i,j\ (t_{ij}\ AD^x\ t_i) \wedge (t_i\ CD\ t_{ij})$

4.2 数据相关

数据相关就是不同事务间的共享数据联系,但此"共享"概念具有比传统的更广的意义,它还包括象嵌套事务模型中的数据继承、通信事务模型的数据通信、合作模型的数据交换、分裂与合并模型的数据"委托".每一事务都有一个与之关联(或它所使用)的数据集,两个事务间的数据相关性就表示它们的数据集的重叠度.形式定义如下:

定义 1. 设 A、B 为两个事务,D_A、D_B 分别为其数据集,A、B 间存在数据相关,当且仅当 $D_A \cap D_B \neq \varnothing$($\varnothing$为空集).

数据相关体现了在事务模型下的数据"可见性"[2],它可以由数据集:存取集、可见集、传递集来说明.

设 t 为一事务,AS_t、VS_t 分别为其存取集、可见集,我们有

定义 2. 事务的存取集就是由它存取的数据集,即 $AS_t = \{d \mid \exists\ p\ p(d) \in t\}$,其中 $p(d)$ 表示数据 d 的操作 p.这里隐含地引用了"事务是数据操作集"的定义.

定义 3. 事务的可见集是它的存取集和所有与它数据相关的事务的存取集的并集.即令 Tt 为所有与 t 数据相关的事务集合,则 $VS_t = \{d \mid d \in (AS_t \cup_{t_i \in T_t} AS_{t_i})\}$

通过存取集间的数据移动,一个事务可以传递数据到另一事务.我们有

定义 4. 由事务 t_1 的存取集移动到事务 t_2 的存取集的所有数据对象的集合称为 t_1 到 t_2 的传递集,记为 $TS(t_1,t_2)$.

事实上,数据传递是上述嵌套事务间的数据继承、通信事务间的数据流、合作事务间的数据交换、分裂/合并事务间的数据委托等的概称.下面分别论述各种复杂事务模型的数据相关性,其中有关表示符的假设同上.

(1)嵌套事务的数据相关

© 中国科学院软件研究所 http://www.jos.org.cn

$\forall\ t_c \in T_c\ VS_{t_c} = AS_{t_c} \cup \{\cup AS_{t_p} | t_c\ AD^*\ t_p\}$

$\forall\ t_c \in T_c\ TS(t_c, t_p) = AS_{t_c}$(传递在 t_c 提交时发生)

〈2〉分裂事务的数据相关

(a)独立分裂:

$AS_{t_b} = AS_t - TS(t, t_c)$

$TS(t, t_c) = AS_{t_c}$(传递在 t 分裂时发生)

(b)顺序分裂:设 t_b 先于 t_c,则

$AS_{t_b} = AS_t - TS(t, t_c)$

$TS(t, t_b) = AS_{t_b}$(传递在 t 分裂时发生)

$TS(t_b, t_c) = AS_{t_b} \cap VS_{t_c}$(传递在 t_b 提交时发生)

$VS_{t_c} = \{d | d \in (AS_{t_c} \cup AS_{t_b}\}$

$VS_{t_b} = VS_t$

〈3〉合并事务的数据相关

$TS(t_a, t) = AS_{t_a}$

$TS(t_b, t) = AS_{t_b}$(传递在合并时发生)

$AS_t = AS_{t_a} \cup AS_{t_b}$

〈4〉通信事务的数据相关

$TS(t_s, t_r) = \{d | d \in AS_{t_s} \cap VS_{t_r}\}$(传递在 t_s 发送数据时发生)

〈5〉合作事务的数据相关

$\forall\ i, j\ TS(t_{ij}, t_i) = AS_{t_{ij}}$(传递在子事务提交时发生)

$\forall\ i, j\ VS_{t_{ij}} = \{AS_{t_i} | t_{ij}\ AD^*\ t_i\}$

4.3　行为相关

事务间的行为相关性是由事务的数据相关性及在共享数据对象上的交互作用而引起的.不象结构相关是直接的,它是由于在同一对象上不同事务操作间的同步所建立的一种间接相关性,通常用事务的"冲突关系"来表示.

定义 5. 若两个操作 p, q 对数据对象 d 所产生的结果输出或结果状态依赖于它们的执行顺序,则称它们在 d 上是"冲突"的,记为 $CT(d, p, q)$.

按此定义,两个操作 p, q 在对象 d 上有冲突关系,当且仅当

$$S(S(s,p),q)) \neq S(S(s,q),p)) \vee R(s,q) \neq R(S(s,p),q) \vee R(s,p) \neq R(S(s,q),p)$$

其中 $R(s,p), S(s,p)$ 分别表示操作 p 对给定状态 s 所产生的结果输出和结果状态.

定义 6. 若两个事务 t_i 和 t_j 分别包含了(至少)两个有冲突关系的操作,则称这两个事务间有(二元)冲突关系,记为 $t_i\ CR\ t_j$.

根据此定义,对任何 $t_i \neq t_j, t_i\ CR\ t_j$,当且仅当

$$\exists\ d \exists\ p, q\ (p(d) \in t_i \wedge q(d) \in t_j \wedge CT(d, p, q))$$

在复杂事务模型下,数据一致性要求可由行为相关性冲突关系来表示[2].

4.4　时间相关

时间相关性是实时数据库事务所特有的,它表明事务的执行顺序或紧迫度,通常以"事务事件"[2]来表示.我们分别以 $BEGIN_t$, $COMMIT_t$ 和 $ABORT_t$ 表示事务 t 的"开始","提

© 中国科学院软件研究所 http://www.jos.org.cn

交”和“夭折”事件的发生时间，两个事务 A,B 的时间相关性用下列模型描述：

$$EXP(TE_A,TE_B,RO,LO);TE_X\subseteq\{BEGIN_X,COMMIT_X,ABORT_X\}(X=A,B)$$

$$RO\subseteq\{=,\leqslant,\geqslant,<,>\};\ LO\subseteq\{\wedge,\vee,\rightarrow\}$$

其中 $EXP()$ 为“时间关系”表达式，TE_A,TE_B 为其运算对象，$\subseteq$ 为“包含于”，LO 为一般逻辑运算符，$=,\leqslant,\geqslant,<,>$ 为“时间关系”运算符，分别表示“*at*”，“*not−after*”，“*not−before*”，“*before*”和“*after*”.

时间相关性可分成两类：

(1)时序相关

事务间按相对关系有 7 种最基本的时序相关，如图 2 所示.

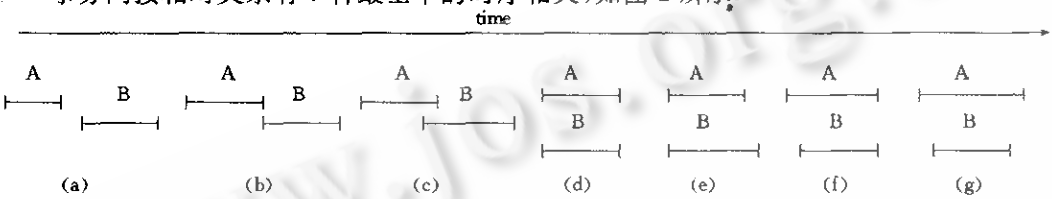

图2 相对时序关系

这些时序相关可按上述模型表示为：

(a)B *after* A：$BEGIN_B>e_A,e_A\in\{COMMIT_A,ABORT_A\}$

(b)B *meet* A：$BEGIN_B=e_A$

(c)B *overlaps* A：$(BEGIN_B>BEGIN_A)\wedge(BEGIN_B<e_A)\wedge(e_B>e_A),e_B\in\{COMMIT_B,ABORT_B\}$

(d)B *equals* A：$(BEGIN_B=BEGIN_A)\wedge(e_B=e_A)$

(e)B *starts* A：$BEGIN_B=BEGIN_A$

(f)B *finishes* A：$e_B=e_A$

(g)B *during* A：$(BEGIN_B>BEGIN_A)\wedge(e_B<e_A)$

2. 带时限的时序相关

上述时序相关仅表明有关事务中相应事务事件发生的时间顺序，没有截止时间要求. 下面举例说明事务间带时限的时序相关.

例 1：事务 A 必须在事务 B 提交后 5 秒内提交. A,B 间的时序相关为

$$(COMMIT_B<COMMIT_A)\wedge(COMMIT_A\leqslant COMMIT_B+5s)$$

例 2：事务 t_1 在事务 t_2 开始后 10 秒内执行. 其时序相关就是

$$(BEGIN_{t_1}\geqslant BEGIN_{t_2})\wedge(e_{t_1}\leqslant BEGIN_{t_2}+10s),e_{t_1}\in\{COMMIT_{t_1},ABORT_{t_1}\}$$

5 事务正确性

在实时数据库情况下，事务的正确性概念比传统的有很大的扩展，它包括：

(1)结果正确性 在实时数据库系统中，由于“时间一致性”[1]要求使事务结果正确性的概念发生了很大变化，需要引入“正确性程度”的概念，依此而分结果正确性为两类：

绝对正确——事务结果反映了正确(一致)的数据库状态.

相对正确——事务结果不是绝对正确的，但在距绝对正确状态允许的一定范围内，因而

© 中国科学院软件研究所 http://www.jos.org.cn

也可算作是正确的.

例如，只读外部有效期[1]内的数据的事务具有 100%的正确度，而读超外部有效期的数据的事务将产生 0%的正确度. 具有硬时间一致性的数据要求有关的事务要具有绝对正确性的结果，而对于具有软时间一致性的数据，相关的事务则可具有相对正确的结果.

(2)行为正确性　这里的"行为"仅指数据操作，行为正确性是与事务的数据存取限制有关的一种事务行为特性. 它由确保事务间行为相关性来实现，即必须解决各事务的冲突关系以提供数据库的一致性. 这就是并发控制所要考虑的.

(3)结构正确性　结构正确性实质上是另一种事务行为特性，它由满足所要求的事务结构相关性来实现. 这是调度与并发控制所要考虑的另一方面.

(4)时间正确性　事务的时间正确性是第三种事务行为(时间行为)特性，它处理一个事务何时可以/必须开始、提交或夭折等(如 4.4 节所述)，故而对实时数据库事务特别重要. 保证事务的时间正确性也就是使事务的定时限制能满足，这就是实时数据库事务调度策略和并发控制机制所要解决的核心问题.

参考文献

1　刘云生. 实时数据库系统. 计算机科学，1994，(3)：21－26.

2　刘云生等. 实时数据库系统的事务特性. 见：徐秋元编，第 11 届全国数据库学术会议论文集，西安：西北工业大学出版社，1993. 464－470.

3　Yungho leu *et al*. Specification and execution of transactions for advanced database applications. Information System, 1992, **17**(2): 171－183.

4　Kopetz H *et al*. Distributed fault－tolerant real－time systems: the MARS approach. IEEE Micro, 1989, **9**(1): 25－40.

5　Wedekind H, Zoerntlein G. Prefetching in real－time database applications. ACM, 1986. 215－221.

6　Korth H F *et al*. Triggered real－time database with consistency constraints. Proc. Int. Conf. VLDB, 1990. 674－698.

7　刘云生，卢炎生. 实时数据库特征及其与主动数据库的联系. 计算机工程与应用，1993，(3)：38－43.

ON REAL－TIME DATABASE TRANSACTIONS

Liu Yunsheng

(*Department of Computer Science and Engineering, Huazhong University of Science and Technology, Wuhan* 430074)

K. Ramamritham　J. Stankovic

(*Department of Computer Science, University of Massachusetts, Amherst, MA* 01003, *USA*)

Abstract　Real－time database transactions show greatly different from traditional database transactions. The differences are mainly displayed in aspects of complex structures, timing property, dependencies and correctness of transactions. This paper first analyzes the application properties and the requirements of real－time database transactions,

© 中国科学院软件研究所 http://www.jos.org.cn

and presents a uniform framework of various complex transaction structures, and then discusses with emphasis on a variety of dependencies between real—time transactions: structural dependency, data dependency, behavior and temporal dependency, as well as real—time transaction correctness in result, behavior, structure and temporal behavior.

Key words Database, real—time database, transaction, real—time transaction, complex transaction, transaction correctness.

© 中国科学院软件研究所 http://www.jos.org.cn