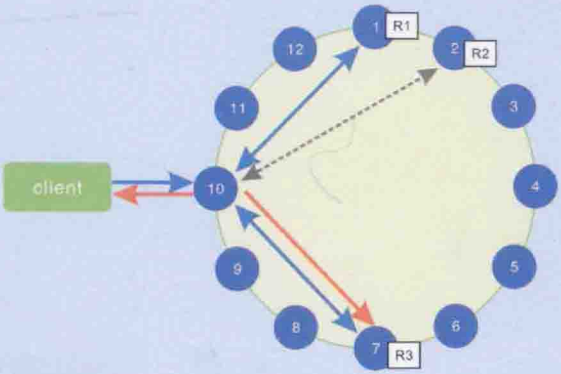
数据的时间一致性维护策略

SHUJUN DE SHIJIAN YIZHIXING WEIHU CELUE

白天 著

中南大学出版社
www.csupress.com.cn

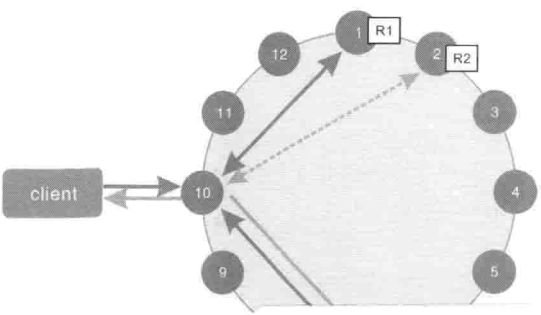


数据的时间一致性维护策略

SHUJUN DE SHIJIAN YIZHIXING WEIHU CELUE

白天 著

中南大学出版社
www.csupress.com.cn



内容提要

本书围绕实时数据库领域中的一个重要问题——数据的时间一致性维护问题展开研究和探讨。在介绍国内外研究现状的基础上，本书着重研究了传感器事务调度、实时导出数据的计算、基于服务质量的事务调度等问题。

本书内容详实、结构严谨、可读性强，既可作为数据库相关专业课程的教学或参考书籍，也可供实时数据库及相关领域的研究、开发人员参考。

图书在版编目(CIP)数据

数据的时间一致性维护策略/白天著. —长沙:中南大学出版社,2015.3
ISBN 978-7-5487-1396-8

Ⅰ.数... Ⅱ.白... Ⅲ.数据库-软件维护 Ⅳ.TP311.13

中国版本图书馆 CIP 数据核字(2015)第 051066 号

数据的时间一致性维护策略
白天 著

□责任编辑 谢贵良
□责任印制 易建国
□出版发行 中南大学出版社
社址:长沙市麓山南路 邮编:410083
发行科电话:0731-88876770 传真:0731-88710482
□印 装 长沙市宏发印刷有限公司

□开 本 880×1230 1/32 □印张 4.25 □字数 120 千字
□版 次 2015 年 3 月第 1 版 □2015 年 3 月第 1 次印刷
□书 号 ISBN 978-7-5487-1396-8
□定 价 25.00 元

前　言

近年来，实时数据库系统在工业过程控制、证券交易、智能交通、军事指挥等领域得到了广泛应用。与传统数据库系统不同的是，实时数据库系统中的数据对象通常具有时间一致性要求：不仅单个数据对象的实时性需要得到有效维护，而且当数据对象间存在关联时，关联对象集上的相互一致性约束也需得到满足。除数据的时间一致性要求外，系统中的事务也具有实时性要求，通常体现为事务的截止期约束。当前学术界对于实时数据库系统的研究多集中于事务的实时性维护方面，在如何有效的维护数据的实时性、如何根据用户需求与系统特点来同时维护两类实时性等问题上的研究则比较欠缺，而这些问题对于实时数据库系统能否为上层应用提供正确和有效的服务至关重要。本书将对这几个问题做一些研究与探讨。全书共分5章，具体内容如下：

第1章，介绍了映象数据和导出数据实时性的具体含义，总结了国内外在数据实时性和事务实时性维护方面的研究现状。

第2章，研究了映象数据的实时性维护。在分析了现有的几种主要调度算法，即HH、ML与DS－FP算法的基础上，提出一种使用时间片交换的延迟调度算法DS－EXC。进而对DS－EXC算法进行了扩展，研究了多处理机上的DS－EXC算法与执行时间可变情况下的DS－EXC算法。

第3章，研究导出数据的实时性维护。具体包括两个方面：基于一些应用中根据部分最新的依赖数据对象值即可计算出满足用户要求的导出对象值的特点，提出不完全导出更新的概念及基

于此概念的导出计算策略，从而扩展了传统的按需更新策略；在映象数据的实时性已得到保证的基础上，研究了如何调度实时查询集，使其中所有查询均可访问相互一致性数据，且能在截止期之前完成或完成总时间最小的问题，给出了最优算法和启发式算法。

第4章，研究了实时数据对象数目或参数动态变化时的实时性维护问题，提出了多处理机传感器事务调度算法的准同步和完全同步切换策略。

第5章，研究了动态负载环境下数据与事务实时性的同时维护问题，提出了一种基于服务质量的策略 UTDS－FC。策略通过数据质量调整，接纳控制与反馈控制来满足用户制定的数据质量与事务质量。进而考虑到不同用户对服务质量的不同要求，提出了多用户事务类下的 UTDS－FC 策略。

本书在撰写过程中得到了刘云生教授、彭超群教授的关心和指导，也得到了湖南理工学院计算机学院提供的帮助和支持，在此一并表示衷心的感谢！由于作者水平所限，书中难免存在错误及疏漏之处，敬请各位同行专家批评指正。

目 录

第1章 绪 论

1.1 数据时间一致性维护概述

实时数据库系统是实时信息服务系统，例如证券交易系统、军事指挥系统，工业控制系统等的核心。其主要功能是实时处理访问实时数据的用户请求[1-3]。更具体地说，实时数据库系统的主要任务有两个：一是维护数据的时间一致性(即实时性)，系统通过接收外界数据源发送的更新数据或主动采样外部对象来更新相应实时数据对象的当前值，以确保它们能真实反映外部环境对象的当前状态；二是确保事务的实时性，系统必须以合理的方式来调度执行用户请求(即用户事务)，以满足这些事务的实时性要求(例如，确保事务在给定的截止期之前完成)。实时数据库系统通常难以同时维护这两方面的实时性，其主要原因有以下几点：

(1)许多系统中存在大量的映象数据，且数据的更新频率较高。例如证券交易系统通常需要实时维护数千支股票的价格，这些股票的价格随着用户的交易不断发生变化。此外，一些系统中还存在外部对象数目的动态变化(例如在煤矿环境监控中，由于坑道长度或数目的增加导致传感器节点数目的增加)、数据更新时间的动态变化(例如在无线多媒体传感器网络中，由于监控要求的变化而导致所采集图像质量的提高)等情况。

(2)一些系统还需维护大量的导出数据。导出数据对象值的计算可能十分复杂，例如在计算股票综合指数时可能涉及标准正

态函数的聚集分布计算。导出数据对象也可能具有高扇入与高扇出，即一个导出对象值的计算需要用到多个其他实时数据对象或非实时数据对象的值，而这个导出对象值又被用到多个其他导出对象值的计算之中。只要依赖集中有一个数据对象的值发生改变，这个导出对象的值就会改变，需要重新计算。

(3)前两点说明系统需付出很大的代价来维护数据的实时性。然而系统同时还需执行用户事务。在实时信息服务系统中，用户事务通常具有比较严厉的时间约束，事务的负载也处于动态变化之中。数据更新与用户事务的同时执行将造成较大的相互干扰。在系统超载的情况下两者的实时性就更加难以保证。

现有实时数据库方面的研究多集中于用户事务本身的实时性维护，而如何有效地维护数据的实时性，如何综合考虑数据和事务的具体特点来同时满足两种实时性要求，如何在资源紧张的情况来对数据和事务的实时性进行折衷以最大化系统的效益等问题的研究则比较欠缺。本书将在这几个方面作一些研究与探讨。

1.2 映象数据的实时性维护

映象数据记录了外部环境实体的当前状态，例如飞机的当前位置或速度、发动机的当前温度等。映象数据的更新通常由传感器事务(又称更新事务)来完成。

1.2.1 映象数据实时性的含义

外部实体的状态随时间不断变化，映象数据的值也必须不断更新，否则系统将访问到过时的数据，做出错误的决策，给用户带来损失。例如，在证券交易系统中基于陈旧的股票价格作出的交易可能导致用户经济上受损，在工厂运行监控系统中由于无法得到最新的运行数据而导致机器故障等。理想情况下映象数据的

值要与外部状态在任何时刻保持一致，实际情况下前者通常“滞后”于后者，从而带来不一致性。其主要原因包括：

(1)通常外部状态的变化是连续的，而传感器节点的采样只能在离散时间点进行；

(2)采样数据在网络中传输会产生一定的延迟；

(3)系统可能由于正在执行其他事务而无法立刻进行数据更新。

对于一个映象数据对象 X，假定其当前值与外部实体的状态保持一致的最迟时刻为 t_1，定义 X 的年龄为系统的当前时间与 t_1 之差。显然 X 的年龄越大，用 X 的值来代表外部实体当前状态的准确程度就越低。令 $\text{value}(X)$ 表示 X 的当前值，$\text{age}(X, t)$ 表示 t 时刻 X 的年龄，$e(X, t)$ 表示 X 对应的外部实体在 t 时刻真实值的估计值，$u(X, t)$ 表示此估计值的不确定程度。以下几种方法可以用来确定 $e(X, t)$ 和 $u(X, t)$[4]。

1. 基于概率密度函数的方法

令 $m_X(v, t)$ 表示 X 的年龄为 t 时对应外部实体状态值的概率密度函数。图 1-1 给出了 X 的当前值为 10，且外部实体的状态变化为随机游走过程时 $m_X(v, t)$ 的一种可能的形式。由图 1-1 可知，当 X 的年龄增加时，外部实体状态值远离 X 当前值的概率也随之增加。

函数 $m_X(v, t)$ 中的参数 t 为相对时间，也可以使用绝对时间参数来定义概率密度函数。定义 $f_X(v, t) = m_X(v, \text{age}(X, t))$，其中 t 为相对于零时刻的偏移量。在实时数据库中，X 表示为 $(\text{value}(X), f_X(x, t))$。

当用户事务在 t 时刻访问 X，可以用 X 的期望值来表示 $e(X, t)$。即

$$e(X, t) = \int_{-\infty}^{+\infty} x \cdot f_X(x, t)\mathrm{d}x$$

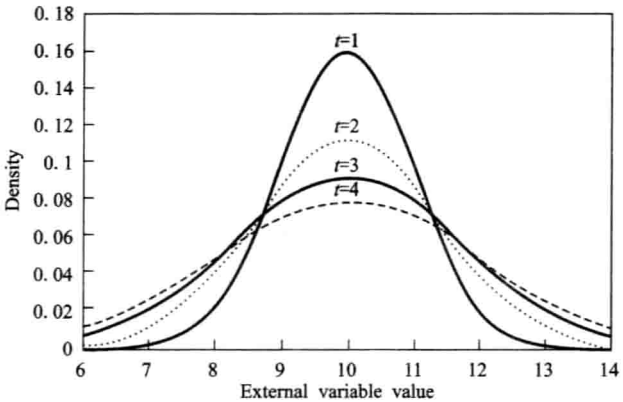


图 1-1 函数 $m_X(u, t)$ 的图像

$e(X, t)$ 的不确定程度 $u(X, t)$ 直接用 $f_X(x, t)$ 来表示。这种方法的问题在于用户事务通常不需要了解 X 的概率密度函数这样具体的信息。另一种方法是使用置信区间。若用户定义的置信水平为 $C(0 \leqslant C \leqslant 1)$，当 $f_X(x, t)$ 为镜像函数时，$u(X, t)$ 由下式确定：

$$\int_{e(X,t)-u(X,t)}^{e(X,t)+u(X,t)} f_X(x, t)\,\mathrm{d}x = C$$

2. 基于有效函数的方法

对函数 $m_X(v, t)$ 进行简化，去掉参数 v，即得到有效函数 $m_X(t)$。简化方法有多种，例如让用户为 $e(X, t)$ 指定一个置信区间，$m_X(t)$ 即为 t 时刻 X 的值落在置信区间内的概率；计算 t 时刻 X 的方差作为 $m_X(t)$ 等。图 1-2 给出使用置信区间方法对图 1-1 中的 $m_X(v, t)$ 进行简化的结果。此时，$m_X(t)$ 的值在 0 与 1 之间。

同样可以使用绝对时间参数来定义有效函数。定义 $v_X(t) = m_X(\mathrm{age}(X, t))$，其中 t 为相对于零时刻的偏移量。在实时数据库中，X 表示为 $(\mathrm{value}(X), v_X(t))$。由于在数据库中没有 X 的分

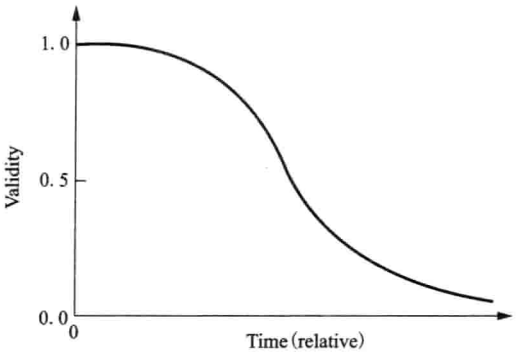


图1-2 有效函数 $m_X(t)$

布信息，因此只能使用 value(X)来代表 $e(X, t)$。$e(X, t)$的不确定性可以简单地通过 $v_X(t)$来表示，即 $u(X, t) = 1 - v_X(t)$。

3. 二元有效函数

如图1-3所示，选取有效函数值的阀值 δ，对应的 X 的年龄称为 X 的绝对有效期，记为 $avi(X)$。当 X 的年龄不大于 $avi(X)$ 时，认为 X 的当前值完全有效，否则认为 X 的当前值完全无效。由此消除有效函数中的不确定性，将其简化为二元函数。

在实时数据库中，X 表示为三元组（$value(X)$，$ts(X)$，$avi(X)$）。其中 $ts(X)$ 为 $value(X)$ 的采样时刻。$avi(X)$ 的值由系统管理员根据数据对象的变化模式、具体应用对数据陈旧度的要求等来设置。

定义1-1 实时数据对象 X 是有效的，或时态一致的，若 $ts(X) + avi(X) \geqslant t_c$。$t_c$表示系统的当前时间。

二元有效函数，即数据的绝对一致性，是时域范围内数据实时性定义的主要形式。相比于概率密度函数和一般有效函数，其定义更简单更直观，便于系统设计者使用。其他的实时性定义将在下文中陆续给出。

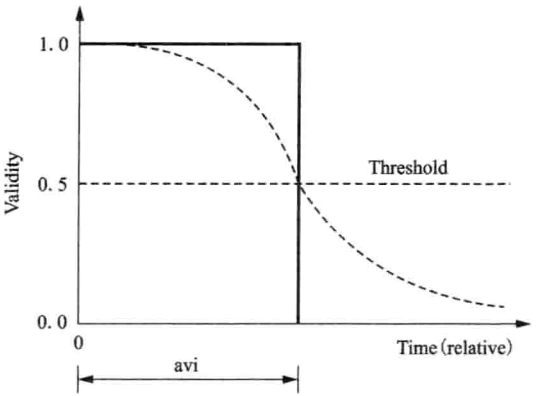


图 1-3 二元有效函数

1.2.2 维护策略

与传统的任务调度算法不同的是，传感器事务的调度需使得数据的时态一致性得到满足。传感器事务调度算法分为基于固定优先级的算法与基于动态优先级的算法。前者主要包括 Half - Half(HH)，More - Less(ML)与 DS - FP。HH[5]与 ML[6,7]分别根据速率单调算法(RM)和截止期单调算法(DM)的可调度性条件来设置传感器事务的周期和截止期。文献[7]给出了基于相似性的 ML 算法，该算法通过跳过产生相似性值的更新来延长传感器事务的更新周期，从而减小 ML 的更新负载。DS - FP[8]在维护数据的时态一致性的基础上充分延迟事务实例的开始时间以减小产生的更新负载。后者主要利用早期截止期优先策略(EDF)来调度事务。事务的截止期与周期根据 EDF 的可调度性充分条件和充要条件来确定[9]。为了减小截止期与周期的计算开销，Li 等提出了一种两阶段算法 $GE_{EDF}^{[10]}$。对于执行时间远小于时态有效期的事

务集合(在实际应用中这类事务集大量存在), GE_{EDF}能在近线性的时间内求出各事务的周期和截止期; 对于其他的事务集, GE_{EDF}使用 ML 算法来进行周期和截止期的计算。

同时具有最大距离约束(即数据时态一致性约束)和最小距离约束(即事务某个实例的完成时间与下一个实例开始时间之差不小于预先定义的阈值)的传感器事务调度问题 (MinMax 问题)由 Zhu 等提出, 用于解决场总线控制系统中的数据采样问题[11]。Zhu 等考虑了分别使用周期性任务模型(ML 与 EDF)和 pinwheel 模型来解决最大或最小值问题。

文献[12]与[13]中给出了一种数据实时性定义并给出更新周期的计算方法。假定实时数据 X_i所对应的外部对象每 O_i时间变化一次, 传感器事务 τ_i周期性更新 X_i的值, 则 X_i的实时性定义为$\frac{O_i}{T_i}$。X_i的权重为 w_i, 代表数据的重要程度。显然, T_i越接近 O_i, X_i的值越能准确反映外部对象的状态。对于一个实时数据对象集, 算法计算每个事务的周期使得数据加权实时性最大同时满足事务集可调度, 即算法解决如下的连续背包问题:

$$\max \sum_{1 \leqslant i \leqslant m} w_i \cdot \frac{O_i}{T_i}$$

$$\text{满足:} \sum_{1 \leqslant i \leqslant m} \frac{C_i}{T_i} \leqslant m \cdot (2^{1/m} - 1)$$

文献同时考虑了另一相关问题, 即在均匀实时性约束下($\forall i, j, \frac{O_i}{T_i \cdot w_i} = \frac{O_j}{T_j \cdot w_j}$)的更新周期计算问题。与 ML 等算法不同的是, 这里的调度算法将产生较大的更新负载, 故不利于用户事务的调度执行。

在 Web 数据库领域, B. Urgaonkar 等提出一种动态实时数据更新策略 LIMD 用于维护数据源与代理服务器缓存数据的一致性

问题[14, 15]。记数据对象 a 在时刻 t 在数据源与代理服务器上的值分别为 S_t^a, P_t^a, 则 a 满足 Δ_v一致性, 若:

$$\forall t,\ |S_t^a - P_t^a| < \Delta_v$$

此处 Δ_v一致性可视为值域范围内的数据实时性定义。策略根据数据值的变化来决定下次访问数据源的时间。当连续两次采样数据值相同时, 下次采样的时间间隔 TTR 将增加到 TTR · $(1+l)$; 当某次采样数据值改变且与前一次采样的时间间隔大于 Δ_v时, TTR 将减小到 TTR · m, 否则 TTR 将增加到 TTR · $(1+\varepsilon)$。l, m 与 ε 为系统参数。在 Web 应用中, 许多数据通常在大部分时间内保持不变, 故策略能有效地减少系统更新负载。若数据值经常处于动态变化中, 策略产生的负载将近似于固定 TTR 策略。另外, 策略并不能保证在任何时刻数据都是一致的。

Web 数据源的数据值通常呈非线性变化。为了利用这一特点来更好的确定 TTR, B. Urgaonkar 等提出一种基于动态模式匹配 TTR 算法(AMPT)[16]。令 $y = \{y_1, \cdots, y_n\}$ 为一个 TTR 序列, 其中 y_n为最近的一个 TTR。$\delta_k = \{\delta_{n-k}, \cdots, \delta_{n-1}\}$ 为差分向量, $\delta_{n-i} = y_{n-i+1} - y_{n-i}$, $1 \leqslant i \leqslant k$。$b_k = \{b_{n-k}, \cdots, b_{n-1}\}$ 为方向向量, 当 y_{n-i+1}大于、小于、等于 y_{n-i}时, b_{n-i}分别取 1、2、0。δ_k 与 b_k合称为大小为 k 的模式。算法首先寻找 $\delta_j = \{\delta_{j-k}, \cdots, \delta_{j-1}\}$, $j < n$, 满足 $b_{j-i} = b_{n-i}$, $1 \leqslant i \leqslant k$。且 j 的取值使得 $\sum_{1 \leqslant i \leqslant k} w_i(\delta_{n-i} - \delta_{j-i})$ 最小。然后算法根据 b_j的取值来计算 y_{n+1}, 其中 $\beta = (1/k)\sum_{1 \leqslant i \leqslant k} \delta_{n-i}/\delta_{j-i}$。

$$y_{n+1} = \begin{cases} y_n + \beta \cdot \delta_{j+1}, & b_{j+1} = 1 \\ y_n - \beta \cdot \delta_{j+1}, & b_{j+1} = 2 \\ y_n, & b_{j+1} = 0 \end{cases}$$

此外 B. Urgaonkar 等还研究了使用反馈控制来动态调整采样的间隔[17]。

文献[18]与[19]研究了如何同步远端数据源及其本地数据库副本。对于远端数据源的 N 个数据对象及对应存放在本地数据库 S 中的 N 个副本，定义 S 在时刻 t 的新鲜度 $F(s, t)$ 如下：

$$F(s, t) = \frac{1}{N}\sum_{i=1}^{N} F(e_i, t)$$

其中 $F(e_i, t)$ 为数据副本 e_i 在时刻 t 的新鲜度。图 1-4 给出了一个数据对象副本 $F(e_i, t)$ 的变化情况。若 e_i 在 t 时与远端对象一致，则 $F(e_i, t)$ 为 1，否则为 0。S 的平均新鲜度定义为 $\overline{F}(S) = \lim_{t\to\infty}\frac{1}{t}\int_0^t F(s, t)\,\mathrm{d}t$。

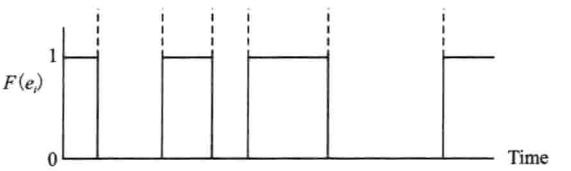


图 1-4 $F(e_i, t)$

远端数据的变化模拟为变化率为 λ 的泊松过程，并由此确定数据副本的同步频率(即单位时间内同步的总次数 f)、资源分配(即 e_i 在单位时间内的同步次数 f_i，$\sum_i f_i = f$)、同步次序(即哪个数据先同步)及同步点(即每次同步在单位时间内的何时进行)，从而最大化 S 的平均新鲜度。文中数据副本的更新时间被忽略不计。文献[18]指出，固定顺序同步优于随机同步与半随机同步，均匀资源分配策略(即 $\forall i, j, f_j = f_i$)优于比例分配策略(即 $\forall i, j, \frac{\lambda_i}{f_i} = \frac{\lambda_j}{f_j}$)。

1.3 导出数据的实时性维护

若计算实时数据对象 X_j 的值时需要用到其他实时数据对象的值，则称 X_j 为导出数据对象，需要用到的其他数据对象的集合称为 X_j 的直接依赖集，记为 $D(X_j)$。X_j 可表示为 $f(X_1, \cdots, X_n)$，其中 $X_k \in D(X_j)$，$1 \leqslant k \leqslant n$。一个包含映象数据对象与导出数据对象的实时数据对象集合可表示为一张有向无环图 $G(V, E)$。V 中每个结点代表一个实时数据对象。若 $X_k \in D(X_j)$，则边 $(X_k, X_j) \in E$。在 G 中 $D(X_j)$ 为与 X_j 有入边相连的结点集合。映象数据对象在 G 中为没有入边的结点。

导出数据对象 X_j 的估计值 $e(X_j, t)$ 与不确定性 $u(X_j, t)$ 可通过 $D(X_j)$ 与函数 f 得到[4]。当函数 f 的形式比较简单时，可以通过 $D(X_j)$ 中数据对象的概率密度函数来推导出 X_j 的概率密度函数，进而得出 $e(X_j, t)$ 与 $u(X_j, t)$。例如，$X_j = X_1 + X_2$ 且 X_1 与 X_2 相互独立时，易知：

$$f_{X_j}(x_j, t) = \int_{-\infty}^{+\infty} f_{X_1}(x, t) f_{X_2}(x_j - x, t)\,\mathrm{d}x$$

$$\begin{aligned} e(X_j, t) &= E(X_1) + E(X_2) \\ &= \int_{-\infty}^{+\infty} x_1 \cdot f_{X_1}(x_1, t)\,\mathrm{d}x_1 + \int_{-\infty}^{+\infty} x_2 \cdot f_{X_2}(x_2, t)\,\mathrm{d}x_2 \end{aligned}$$

根据 $f_{X_j}(v, t)$ 及用户给出的置信水平可以求出置信区间作为 $u(X_j, t)$。若 X_1 与 X_2 不独立，则需要定义 X_1 与 X_2 的联合密度函数。例如可以先定义出相对时间联合密度函数 $m_{X_1, X_2}(v_1, v_2, t_1, t_2)$，再得到绝对时间的联合密度函数 $f_{X_1, X_2}(v_1, v_2, t)$，并由此得到 X_j 的概率密度函数如下：

$$f_{X_j}(x_j, t) = \int_{-\infty}^{+\infty} f_{X_1, X_2}(x, x_j - x, t)\,\mathrm{d}x$$