计算机科学 1994 Vol.21 №.6

时态演绎数据库——模型与查询语言

彭 铭 戴 军 招兆铿
(复旦大学计算机科学系 上海 200433)

摘 要 The temporal deductive database model—— TedDM is developed from a deductive database model by applying temporal reasoning method (T-resolution), to delineate the deductive mechanism. The query language called TedSQL, which is a superset of SQL, is proposed to retrieve imformation from temporal relations. TedSQL also provides the temporal reasoning capabilities for deductive mechanism, The design for TedDBMS, a temporal deductive database management system, is made. It is going to be carried out on the deductive database system LIDBS 3.0.

关键词 Temporal reasoning, Temporal database system, Temporal relational algebra.

时态演绎数据库是引入有效时间、事务时间概念的演绎数据库，能比较真实地反映现实世界的现象和发展状况，因此能满足现在的许多应用，如规划、决策支持系统、医疗信息系统等，因为这些应用不仅需要当前的数据，也需要历史数据，要求系统能处理时间信息。同时，时态数据库与人工智能领域有着密切的关系。例如，股票分析系统根据股票行情的历史信息进行预测，机器人系统需要确认机器人完成一系列任务的期限是否满足要求等，都要用时态信息进行推理，即所谓的时态推理(TR)。这就需要时态逻辑作为理论基础，保证系统的正确有效性，并可作为系统性能评价的基准。另外，时态数据库与时态推理的有机结合，引出了时态演绎数据库、时态知识库的新课题。

本文将从介绍一种基于片段的时态逻辑以及这个形式系统的推理方法—T-归结出发，分别讨论时态演绎数据库的模型定义和查询语言设计的问题。最后对我们的工作做一个评价。

一、数据模型

1.1 时间模型

时态数据库一般涉及到以下三类时间：(1)有效时间：是指一个对象在现实世界中存在的时间；(2)事务时间：是指一个对象进行数据库操作的时间；(3)用户自定义时间。

有效时间和事务时间是抽象出来由系统统一处理的时间。时态数据库的目的就是将原本由用户处理的时间由系统统一处理。

1.1.1 时间的表示 有关时态系统中时间的表示有多种观点。我们采用时间点作为原始时态概念，用有序时间点对来定义时间片，作为时间片的始、终端，如果始、终端相同，则时间片退化为一个时间点。时间是无限的，每个时间均有前导和后继。时间是线性的，两个时间点之间存在一种全序关系。

另外，我们采用时间是离散的观点，首先时间的计量与采用的时间粒度有关，不能做到完全精确真实；其次用时间点和时区的概念可以比较自然地描述那些发生在某一时期而不是瞬间的事件；还有考虑到实现时总要对时间进行存贮，与离散模型更为贴切。与此相关，模型中有效时间和事务时间各有其一致的时间粒度，不允许嵌套。这种处理使得时间的管理比较直截了当，但又不失其一般性。当然，这要得到用户的支持。

有了以上几方面的认识，可以看出，时间与整数 Z 是同构的。

1.1.2 时间运算 为了方便以后在时态演绎数据库中描述时态运算，我们定义一下时间的运算。

设(TW, ≤)为时间结构，a, b, c, d∈TW，则[a, b]就定义了一个时区 i，i 包含了所有 a 到 b 的时间点。令 NULL=[d, c]指 c<d。以下的时态比较运算产生 Boolean 值：

(1)属于 in：

c in [a, b] iff (a≤c≤b)

(2)包含 during：

彭铭 硕士生，研究方向为逻辑及其应用。戴军 硕士生，研究方向为时态演绎数据库。招兆铿 教授，研究方向为人工智能和自动推理。

[a,b] during [c,d] iff(a≥c∧b≤d)

(3)相等 equal:

[a,b] equal [c,d] iff(a=c∧b=d)

(4)覆盖 overlap:

[a,b] overlap [c,d] iff(a≤d∧c≤b)

(5)后继 follows:

[a,b] follows [c,d] iff(a=d+1)

(6)前导 precedes:

[a,b] precedes [c,d] iff(c=b+1)

(7)邻接 adjacent:

[a,b] adjacent [c,d] iff(c−b=1∨a−d=1)

(8)先于 before:

[a,b] before [c,d] iff(b<c)

(9)后于 after:

[a,b] after [c,d] iff(a>d)

可以看出,其中只有 equal,overlap,adjacent 运算是可交换的。

以下运算产生新的时区:

(1)并(+)

[a,b]+[c,d]=

[min(a,c), max(b,d)]	或	如果[a,b]overlap[c,d] [a,b]adjacent[c,d]
NULL		否则

(2)交(*)

[a,b]*[c,d]=

[max(a,c), min(b,d)]	如果[a,b]overlap[c,d]
NULL	否则

(3)差(−)

[a,b]−[c,d]=

[a,b]	如果[a,b]before[c,d]∨[a,b]after[c,d]
[a,c−1]	如果[a,b]overlap[c,d]∧c in [a,b]
[d+1,b]	如果[a,b]overlap[c,d]∧a in [c,d]
[a,c−1]∪[d+1,b]	如果[c,d]during[a,b]∧~([c,d]equal [a,b])
NULL	否则

注意,时区的差的运算在此不封闭,结果可能出现前后两个不相邻的时区,我们暂且用"∪"来表示此结果,到后面再加以讨论。

现在许多研究一般都基于关系数据模型并扩充以时态概念。Snodgrass[10]根据数据库中使用的三类时间将数据库分为四种:快照数据库、滚回数据库、历史数据库和时态数据库。参考此分类观点,我们认为时态数据库中,有效时间和事务时间应同时得到支持;而且,有效时间是本质的内容,与数据库模型有着密切的关系,而事务时间则在系统事务活动中才起作用。所以本章涉及时态概念时,着重讨论有效时间的处理。

首先我们建立一个时态数据库模型。

1.2 时态数据库模型

我们提出一个时态数据库模型,采用基于片段的时区标记法;同时,为了消除时态异常,引入了时态范式的概念。

设域为 DOM=W,时态域为 TDOM=TW。给定一组域 D1,D2,…,D_n 及时态域 TD1,TD2,其笛卡儿积为 D1×D2×…×D_n×TD1×TD1×TD2×TD2={(d1,d2,…,dn,td1,td2,td3,td4)|di∈Di;td1,td2∈,TD1,td3,td4∈TD2},其中每个元素(d1,d2,…,dn,td1,td2,td3,td4)称为一个元组;一个时态关系 R 即为上述元组的集合,并且 R=D1×D2×…×Dn×TD1×TD1×TD2×TD2。

设 A:D1×D2×…×Dn×TD1×TD1×TD2×TD2→D1×D2×…×Dn;TA:D1×D2×…×Dn×TD1×TD1×TD2×TD2→TD1×TD1,则 A(R)可以看作为一个表格,其中的列称为属性,TA(R)表示 R 的有效时间,TD2×TD2 是 R 的事务时间。R(A1:D1,A2:D2,…,An:Dn,Valid-From:TD1,Valid_To:TD1,Trans_From:TD2,Trans_To:TD2)称为时态关系模式,时态关系模式也有完整性约束,以保证每个实例都有意义。一个时态数据库模式就是一组时态关系模式与完整性约束的集合。

这里,我们要强调的是时态域 TD1,TD2 的特殊性,对于 Valid_From,Valid_To,Trans_From,Trans_To,我们也不称它们为时态属性。TD1×TD1 和 TD2×TD2 都是一个有序的时间点对的集合,其中 TD1×TD1 与 D1×D2×…×Dn 有本质上的联系,而 TD2×TD2 则与 D1×D2×…×Dn 在事务活动中才发生关系。而且,一个时态关系除了保留传统关系中的一些特征外,为保证系统的一致性,还应使每个元组标记的时区是最大的,不可再合并。另外,尽管时态关系在传统关系上扩充了两个时间维,但我们这里仍称其为表。

在模型中，每个元组都有一个确定的 VALID_FROM 的值，而 VALID_TO 的值除确定的值以外可以是以下两种：

$$VALID\_TO=\begin{cases}NOW & \text{如果 } VALID\_FROM\leqslant NOW\\ UNKNOW & \text{如果 } VALID\_FROM>NOW\end{cases}$$

对于那些过去发生的事件，VALID_TO 未知时，仅能假定它至少到现在为止为真，在模型中，我们用 NOW 来表示而不用∞，这是因为∞隐含将来也为真，显然与事实不符。对于那些将来才发生的事件，VALID_TO 当然未知，则用 UNKNOW 来表示这个元组将来的真假不可预测。区别以上两种未知值，可以更好地管理时态数据库，并更准确地用来推理。另外，相应于 NOW，我们引入函数 now( )对当前时间求值。

1.3 时态关系的运算

我们提出的模型是对传统关系模型的一个扩充，时态关系的运算就是扩充经典关系运算而来的，增加了一些时态方面的特征。由于我们认为两个不同粒度的时态关系的笛卡尔积是没有意义的，所以我们不支持时间粒度的嵌套，而联接也有了新的内容。

首先，我们给出一个算法 Refine，用来对关系运算的中间结果加以整理，最大限度地合并有效时间时区，消除冗余。

[Refine 算法] 对于某个快照状态：

①如果存在元组 t，TA(t)将出现两个时区 $i_1\cup i_2$ 的形式，则插入元组 t_1,t_2，使 $A(t_1)=A(t_2)=A(t)$，$TA(t_1)=i_1$，$TA(t_2)=i_2$，并删除 t；

②删除所有 TA(t')=NULL 的元组 t'；

③∀ tp，若∃ tq，使 A(tp)=A(tq)并且 TA(tp) overlap TA(tq)或 TA(tp)adjacent TA(tq)则删除 tp、tq，并插入新元组 t，使 A(t)=A(tp)并且 TA(t)=TA(tp)+TA(tq)。

下面我们定义有关时态关系的五种基本运算。

设时态关系：

R1 (A1, …, An, Valid_From, Valid_To, Trans_From, Trans_To)；

R2 (A1, …, An, Valid_From, Valid_To, Trans_From, Trans_To)；

R3 (B1, …, Bm, Valid_From, Valid_To, Trans_From, Trans_To)。

(1)并 Union(∪)。R1 和 R2 的并是 R(A1, …, An, Valid_From, Valid_To, Trans_From, Trans_To)，其中，若 t∈R1 或 t∈R2 则 t∈R'，

R=Refine(R')。

(2)差 Relference(－)。R1 和 R2 的差是 R(A1, …, An, Valid_From, Valid_To, Trans_From, Trans_To)。对∀ $t_1\in R1$：如果∀ $t_2\in R2$ 都有 $A(t_1)\neq A(t_2)$则 $t_1\in R'$；如果∃ $t_2\in R2$ 使 $A(t_1)=A(t_2)$则 $t'\in R'$，并且 $A(t')=A(t_1)$且 $TA(t')=TA(t_1)-TA(t_2)$。

R=Refine(R')

(3)投影 Project(π)。R 是 R1 的投影，$R=\pi_{A\_set1}(R1)$。令 A_set={A1, …, An}，A_set1={Ai, …, Aj}=A_set，投影运算中系统自动指定有效时间 Valid_From 和 Valid_To。可得 R' (Ai, …, Aj, Valid_From, Valid_To, Trans_From, Trans_To)，其中∀ t∈R1，∀ Ak∈A_set1，都有 t(R'.Ak)=t(R1.Ak)。

R=Refine(R')

(4)选择(σ)。$R=\sigma_{con}(R1)$。选择所有那些满足 Con 的 R1 中的元组，Con 中可以包括时态的比较关系。

(5)时态联接 Tjoin(⋉)，R' (A1, …, An, B1, …, Bm, Valid_From, Valid_To, Trans_From, Trans_To)，其中，若 $t_1\in R1$，$t_3\in R3$ 则 $t\in R'$ 并且 $A(t)=(A(t_1)\ A(t_3))$，$TA(t)=TA(t_1)*TA(t_3)$。

R=R1⋉R3=Refine(R')

联接 $R1\underset{A_i\theta B_j}{\overset{T}{\ltimes}}R3=\sigma_{R1.A_i\theta R3.B_j}(R1\overset{T}{\ltimes}R3)$，其中 θ 是任意一种算术比较算子。

自然联接(⋉)是联接运算中的一类，它要求参与运算的两个关系在同名属性上具有相同的值。在产生的结果中，同名属性只出现一次。

若 A1, …, Ak 是 R1 和 R2 的同名属性，则 $R1\overset{T}{\ltimes}R2=\pi_{i1,\cdots,im,R1.A1,\cdots,R1.Ak}\sigma_{R1.A1=R2.A1\wedge\cdots\wedge R1.Ak=R2.Ak}(R1\overset{T}{\times}R2)$，其中，i1, i2, …, im 是 R1⋉R2 的除 R1.A1, …, R1.Ak 外的所有属性。

需要指出的是，以上运算都产生新的时态关系。其中，我们没有给出面向事务时间方面的特定的运算，因为，我们认为，不同层次的历史关系的运算是不必要的，也是没有意义的。除了选择运算 σ 之外，所有作为运算结果的时态关系中每个元组的事务时间 Trans_From 都为 now( )，而 Trans_To 都为 NOW。在时态联接的定义中，我们采用了时区的交的语义，这样会丧失一些历史信息，但从另一角度来看，它具有如下性质：若 R,S,T 为时态关系，则 $R\overset{T}{\ltimes}(S-T)=(R\overset{T}{\ltimes}S)-(R\overset{T}{\ltimes}T)$。而这是文[5]等采用时区的并的语义所没有的。

例 S：

S	V_F	V_T
180	10	20

T：

S	V_F	V_T
180	15	20

R：

E	M	V_F	V_T
52	S	10	30

则 R×(S−T)为：

E	M	S	V_F	V_T
52	S	180	10	14

而(R×S)−(R×T)为：

E	M	S	V_F	V_T
52	S	180	10	14

若用并的语义，则后者为空，而前者变为：

E	M	S	V_F	V_T
52	S	180	10	30

显然两者不相等。

事实上，与传统关系代数相似，在我们定义的时态关系的运算中也支持如下的代数等价式，这些代数等价式将是我们实现查询语言的优化机制的一个依据。

(1)$Q \cup R = R \cup Q$；

(2)$Q \times R = R \times Q$；

(3)$\sigma_{F_1}(\sigma_{F_2}(R)) = \sigma_{F_2}(\sigma_{F_1}(R))$；

(4)$Q \cup (R \cup S) = (Q \cup R) \cup S$；

(5)$Q \times (R \times S) = (Q \times R) \times S$；

(6)$Q \times (R \cup S) = (Q \times R) \cup (Q \times S)$；

(7)$Q \times (R - S) = (Q \times R) - (Q \times S)$；

(8)$\sigma_F(Q \cup R) = \sigma_F(Q) \cup \sigma_F(R)$；

(9)$\sigma_F(Q - R) = \sigma_F(Q) - \sigma_F(R)$；

(10)$\pi_x(Q \cup R) = \pi_x(Q) \cup \pi_x(R)$。

对于(8)和(9)，显然要求 F 中不包括时间的比较关系。对于(10)，则要求 x 中不包括有效时间和事务时间。以上证明可参见文[14]。

1.4 用逻辑的观点来看时态数据库模型

在以上讨论中，我们提出了一个时态关系数据库的定义，它是一组时态关系及其上运算的集合。从逻辑角度看，时态关系数据库是一组谓词的集合，每个谓词是一个时态关系。这样时态关系数据库可以看作是一阶时态逻辑语言的一个实现，谓词演算就是表达时态关系运算和查询的一种形式语言。

1.5 时态演绎数据库模型

对于时态数据库 TDB 的理论 T，TDB 满足完整性约束 W 当且仅当 $T \vdash W$，并且在某个快照状态，对于查询 R(x1,…,xn,tx1,tx2)的回答就是所有这样的元组：{(d1,…,dn,td1,td2)|$T \vdash$R(d1,…,dn,td1,td2)}。

在时态演绎数据库中，我们假定演绎规则是封闭的合式公式，且不出现函数符号，我们仅考虑它们的事务时间，并用如下形式来表示：

$\langle p,ts,te\rangle \leftarrow \langle p1,ts1,te1\rangle \wedge \cdots \wedge \langle pn,tsn,ten\rangle,[t1,t2](n \geqslant 0)$

n＝0 时，即为事实。其中：(p,ts,te)为正文字，(pi,tsi,tei)(1≤i≤n)为文字，[t1,t2]是规则的事务时间。

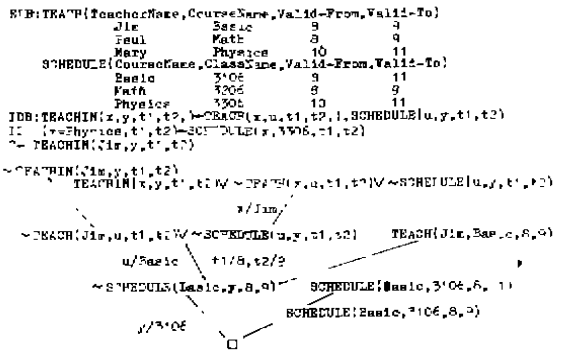
为了从实现的角度来定义时态演绎数据库，我们引入元规则“失败即否定”(NAF)及一个控制机制，控制机制使所有元组都满足时态公理。

我们称时态数据库为外延数据库 EDB，EDB 中关系谓词称为外延谓词。内涵数据库 IDB 由演绎规则组成，都是范围受限的 Horn 子句，即子句中每个出现在头部的变元也出现在体上。这些子句的头部的谓词就称为内涵谓词。

这样，我们定义时态演绎数据库 TDDB＝(EDB,IDB,IC,NAF)。

以上形式化了一个时态演绎数据库。其推理机制可用 T-归结来描述，有关时态逻辑形式系统及 T-归结的内容可参阅文[13,12]，在此我们可以举一个简单的例子。

例 在某个快照状态：



这样，得到 y＝3106，t_1＝8，t_2＝9，即查询结果为 TEACHIN(Jim,3106,8,9)。

至此，我们从关系数据库模型出发，定义了时态数据库模型，又用逻辑的观点形式化了时态演绎数据库模型，并给出了其演绎机制。对于理论上的探讨，本研究小组已作了相当多的工作，包括完整性约束检查方法，T-归结的正确性和完备性的证明，时态演绎数据库中非单调推理等研究[9,13,7,12]。在这里，我们要面向实际应用来开展我们的研究工作，并提出一个切实可行的时态演绎数据库管理系统 Ted-DBMS 的设计方案来。

二、设计方案

由于时态演绎数据库是数据库与人工智能结合的产物，所以其管理系统可能的实现方法有：从时态数据库管理系统出发增加演绎功能，及从逻辑程序设计语言 PROLOG 出发，增加时态数据库管理功能。我们按照一体化的方法进行设计，图 1 给出了时态演绎数据库管理系统 TedDBMS 的总体结构。其中 RDL Compiler 和 DDL Compiler 分别负责处理 EDB 模式和 IDB 模式的定义，并形成 EDB/IDB 描述表；Query Language Processor 接收来自用户或应用程序的查询，并对应用程序中数据操纵语句进行预处理。查询处理器可以访问 IDB/EDB 描述表，根据是否存在索引对查询进行适当优化。Query/Inference Engine 对查询进行求值，并将有关概念级上的命令转化为物理级上，即文件级上的命令。这些命令然后由文件处理器处理，存取物理数据库。

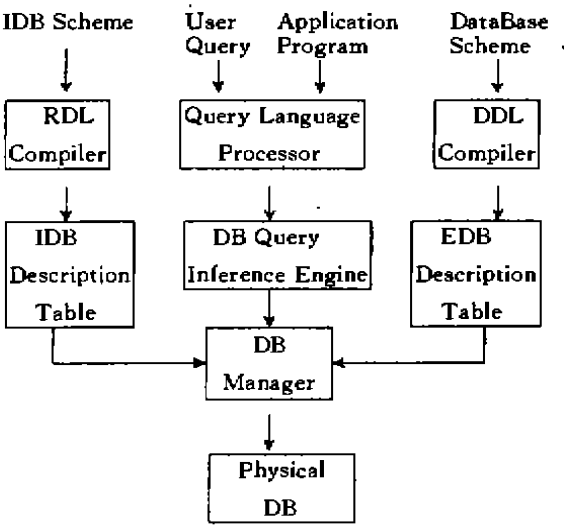


图 1

下面我们将着重研究时态数据库查询语言 TedSQL，尽管我们的模型是四维结构的，但仍然沿用表的概念，并称时态关系模式为时态基表。

2.1 查询语言 TedSQL

查询语言 TedSQL 是时态演绎数据库管理系统的用户界面，用户可以通过终端，以联机方式使用，也可以将命令嵌入主语言，完成查询功能。

TedSQL 是 SQL 的一个子集的扩充，主要增加了时态概念及知识表达能力，TedSQL 除了保留 SQL 语言的基本功能外，还允许用户作历史查询和滚回查询。此外，用户还可以使用一种不出现函数的 Horn 子句来表示从若干已知关系推导出某些关系的规则。系统中的推理查询语句采用了附加推理要求（INFER）的选择语句。在 TedSQL 中，我们用 VALID 子句表示有效时间，用 WHEN 子句表示有效时间之间的时态关系，并引入 AS—OF 子句，通过事务时间来指明某个快照状态，以便进行滚回查询。TedSQL 的语法与 SQL 的语法基本保持一致，可参见文[14]。现在我们讨论一下其时态及演绎方面的功能。

2.1.1 *数据定义* 时态关系模式应该包括四个基本的成份，即 Valid_From，Valid_To，表示有效时间，以及 Trans_From，Trans_To 表示事务时间。

建立时态关系模式，格式为：

CREATE TABLE〈TableName〉(〈ATTR DEF〉{，〈ATTR DEF〉})

语法和 SQL 一致，可以有数据约束 NOT NULL。

通过 CREATE TABLE 语句，系统建立一个时态基表，它具有用户指明的几个属性，以及 Valid_From，Valid_To，Trans_From，Trans_To 表示有效时间和事务时间。我们用 TableName.DURATION 来表示它的有效时间时区：[Valid_From，Vlid_To]。

由于我们的时态数据库是非删除性的，TedSQL 实际上不提供 DROP TABLE 语句。下面先引入几个时态方面的子句，并讨论它们在查询中的运用。

2.1.2 *时态子句*

(1)WHEN 子句。与 WHERE 子句相似，它指出两个有效时间时区的时态关系。WHEN 子句与 WHERE 可以一起使用。

(2)VALID 子句。指明有效时间的开始、结束或时区，可以有以下格式：

VALID FROM〈Valid_From〉|VALID TO〈Valid_To〉|VALID FROM〈Valid_From〉TO〈Valid_To〉

(3)AS—OF 子句。指明事务时间，支持滚回查询，出现在 SELECT 语句中。

以上讨论了几个时态子句，并举了几个查询的例子。下面我们来分析一下数据库更新操作中时态子句的运用。

2.1.3 *数据更新操作*

(1)插入 INSERT。此语句中可以用 VALID 子句指明有效时间，若缺省则为：VALID FROM now( )TO NOW，插入一个元组，其事务时间为 Trans_From=now( )，Trans_To=NOW。

(2)删除 DELETE。此语句中可以用 VALID 子名指明有效时间,可以用 WHEN 子句说明有效时间之间的时态关系作为条件。

DELETE 的处理与传统数据库不同,事实上时态数据库是非删除性的,保留了对象的所有历史情况,以便支持滚回查询。其实现过程如下:①找到一组满足条件且 Trans_To=NOW 的元组;②将它们的 Trans_To 都置为 now( )。

(3)修改 UPDATE。此语句中可以用 VALID 子句指明有效时间,可以用 WHEN 子句说明有效时间之间的时态关系作为条件。修改有效时间则用类似格式:SET Valid_From=t1,Valid_To=2

UPDATE 的处理结果与传统数据库中不同,其实现过程如下:①找到一组符合条件且 Trans_To=NOW 的元组;②复制以上所有元组;③将原先所有元组的事务时间 Trans_To 置为 now( );④修改复制出的所有元组,使其满足更新要求。

以上给出了有关的 TedSQL 语句,并举例说明。如果将有效时间和事务时间当作普通属性,以上语句实际上都可以用 SQL 语言中语句来表出。所以,其形式语义的说明也可以此来加以描述。

至此,我们对 SQL 的扩充作了讨论,事实上 TedSQL 除了强大的数据操纵功能外,还可提供推理功能。

2.2 推理和完整性约束检查

TedSQL 中用语句 SELECT/INFER 来表示查询推理的要求,推理结果的处理则与普通查询一样。下面是有关规则的定义及管理。

(1)创建时态谓词模式 CREATE RULE

CREATE RULE〈RuleName〉(〈ITEM DEF〉{,〈ITEM DEF〉})

谓词符在系统中表示为一个虚关系模式,所以,该语句的功能与建立时态关系模式基本一致,并且,谓词模式一旦建立,即具有 Trans_From 和 Trans_To 标记其事务时间;Valid_From 和 Valid_To 标记其有效时间。

(2)规则插入 INSERT RULE

INSERT RULE〈rule〉

其中 rule 是 DATALOG 的规则。其事务时间为[now( ),NOW],有效时间为[T1,T2]。

(3)规则删除 DELETE RULE。将该规则的事务时间 Trans_To 变为 now( )。虚关系中规则的插入和删除的处理与实关系中元组的插入和删除的处理类似。

已有的研究工作[8,17]推出了用时态关系演算构造 RAE 树实现推理查询的方法,文[13]给出了完整性约束检查的 TSLDNF 的实现算法,从某个角度来看,这实际上是一种计算过程,由于它利用时态关系演算来计算、所以每次计算完成即可得到所有的答案,效率比较高,而且利用 RAE 树可以找到某些优化算法。在我们的模型中这种推理方法是与 T-归结等价的[15]。现在我们正在讨论 T-归结的实现算法,从而系统地实现一个管理系统 TedDBMS。

三、总结

现在对时态数据库的研究正得到越来越多的重视,已有不少文献资料推出了各自的研究成果。McKenzie 和 Snodgrass 综合了现今十几个时态关系数据库的特点,提出了二十六条有关时态关系代数评价的标准[6]。当然,由于各项研究的角度、侧重点不同,其中有些标准是冲突的。我们将针对每一条标准来评价我们的模型 TedDM。原先二十六条标准是按字母序排列的,我们这里仍然按照原来的次序。

(1)不适用。是否均匀模型,该标准涉及 Non-1NF 模型。

(2)部分满足。快照代数的相应扩充。由于时态概念的引入,联接等有了新的内容。

(3)不满足。TedDM 不支持数据的周期性。

(4)每个合适域的合法属性值的组合是一个合法的元组。

(5)任何合法元组的集合可产生合法时态关系。当然,我们所指时态关系都是经过 Refine 操作的。

(6)定义了形式语义。以时态逻辑为基础。

(7)未讨论。关于时态关系演算和时态关系代数的表达能力未作讨论。

(8)未讨论。关于聚集函数,我们假定保留了原有关系代数的聚集函数。

(9)不满足。对增量语义未作定义。

(10)部分满足。对时态关系的交、商未作定义,但可利用已有的运算来定义。

(11)是一个代数。

(12)模型的定义不需要空值。

(13)不满足。TedDM 不支持时间粒度的嵌套。为简化问题,便于实现。

(14)若将有效时间和事务时间看作普通属性,则成为快照模型。

(15)保留 1NF,引入 TNF 概念消除异常。

(16)历史关系及运算的三维模型。TedDM 中笛

卡尔积以时态联接代替，消除与第23条的冲突。

(17)部分满足代数等价性。相应于笛卡尔积的时态联接，我们采用时区的交的语义，对于差的分配律也成立。

(18)不满足。TedDM不支持全部四类关系，只支持时态关系。

(19)支持滚回查询。

(20)不满足。TedDM不支持多重关系模式，即不记录模式随时间变化的历史。

(21)支持静态属性，允许存在不随时间变化的属性。

(22)同时支持有效时间和事务时间，而且，这两条时间维是正交的。

(23)采用元组时区标记法，而非属性时间标记法。

(24)每个时态关系的表出是唯一的。

(25)单类而非多类模型。时态关系运算要求输入都是时态关系。

(26)有明确的更新语义。我们用时态关系演算的方式给出了时态关系运算的语义。

从以上可以看出，我们的系统具有良好的理论基础，又结合了成熟的技术，可操作性好，易于实现；而且，真正同时支持了有效时间和事务时间的概念，又结合了演绎机制，能够比较全面地反映现实世界发展的状况。

参考文献

[1] Ahn, I. et al., Performance Evaluation of a Temporal Database Management System, ACM-SIGMOD, 1986, pp. 96-107

[2] Clifford, J. et al., Formal Semantics for Time in Database, ACM Trans. Database Systems, Vol. 8, No. 2, 1988, pp. 214-254

[3] Clifford, J. et al., On an algebra for historical relation databases: two views, ACM-SIGMOD, Austin, pp. 247-265

[4] Gallaire, H. et al., Logic and Databases: A Deductive Approach, ACM Computing Surveys, Vol. 16, No. 2, 1984, pp. 153-185

[5] Jones, S. et al., Time dimension in a database in Proc. of the Int. Conference on Databases, Hyden, U. K. 1980

[6] McKenzie, E. et al., Evaluation of Relation Algebras, ACM Computing Surveys, Vol. 23, No. 4, 1991, pp. 501-543

[7] Navathe, S. B. et al., A Temporal Relational Model and a Query Language, Inform. Sciences, Vol. 49, 1989, pp. 147-175

[8] Qian, Wei and Zhao, Zhaokeng, Reasoning by RAE Tree Proc. of 6th. National Conference on Database, 1987

[9] Qian, Wei and Zhao, Zhaokeng, Temporal Reasoning Management with Nonmonotonicity, Information Processing 89, Elsevier Science Publishers B. V. North-Holland, pp. 667-672

[10] Snodgrass, R. et al., A taxonomy of temporal databases, Proc. ACM-SIGMOD, Austin, 1985, pp. 236-246

[11] Ullman, J. D. Princ. of Database and Knowledge-base Systems, Vol 1, 1988, Computer Science Press, Inc.

[12] Zhao, Zhaokeng & Dai, Jun Automated Theorem Proving in Temporal Logic: T-resolution, 计算机学报(英文版)1994

[13] 陈文丹(1991)，时态演绎数据库的完整性约束，复旦大学计算机科学系硕士论文

[14] 戴　军(1993)，时态演绎数据库——模型与查询语言，复旦大学计算机科学系硕士论文

[15] 钱　伟(1989)，时态演绎数据库管理系统的研究与开发，复旦大学计算机科学系硕士论文

[16] 施伯乐等(1989)，关系数据库的理论及应用，河南科学技术出版社

[17] 张　科(1992)，非单调时态演绎数据库的研究，复旦大学计算机科学系硕士论文