计算机科学 2000Vol 27№.8

移动对象数据库：问题及其解决方案

Mobile Object Database: the Problems and their Solution

李 东 冯玉才 王元珍

（华中理工大学计算机学院 武汉 430074）

Abstract Mobile object database have a widely application perspective both in military industry such as in digital battlefield and civilian industry such as in transportation systemm. DBMS is the base of database application, but existing DBMS are not used for mobile database application because of lacking some special abilites such as describing dynamical data, different lignustics issue and uncertainty management etc. In this paper, we will discuss the problems in mobile database and the solution to the problems.

Keywords Mobile, Database, Modeling, Query, Uncertainty

一、引言

在记录了移动对象以及位置的相关信息的数据库中，比如有一个存储了出租车位置信息的数据库，对该数据库的一个典型查询是：查询在牌号为 ABC123 卡车（需要他人帮助）周围 1 公里内的卡车；又如，在包含对象位置的战场数据库中，一个典型的查询可以是：检索给定地区的友方直升机，或者检索在这个地区 10 分钟内将到达的友方直升机。这些查询可以发自移动对象也可以发自静态用户，我们将具有以上特性的应用称为移动对象数据库（MOD）应用；进行以上查询称为移动对象查询。

在军用方面，移动对象数据库应用出现在数字战场环境中，在国内工业中出现在交通运输系统中。比如，由美国 Qualcomm 公司开发的 Omnitracs 系统使用在交通运输行业，它就具备了移动对象数据库的功能，提供了位置管理，借助卫星使车辆与数据库连接起来，这些车辆配备了 GPS 系统，可以自动、定期报告它们的位置。

DBMS 技术是开发移动对象应用的底层支持，但是现有的 DBMS 并不是针对移动对象应用来开发的。其原因是移动对象数据库应用中需要一些特殊能力，而在现有 DBMS 中不能提供这些能力。支持移动对象数据库应用所需的特殊能力有：

• 位置建模 现有 DBMS 系统并不能很好地处理连续变化的数据，如移动对象的位置。其原因是：数据库系统中的数据是不变的，除非它被修改。比如查询数据库中的移动用户（例如车辆）及它们的位置，车辆的位置数据由于其移动而发生连续修改。这显然是不能令人满意的，要么位置修改过于频繁（这会严重影响系统的性能），要么查询的结果过时了。

• 语义问题 通常，在移动对象应用中的查询都要涉及空间对象（例如，点、线、面、多边形等）及时间条件，比如，考虑这样的查询："查询将要达到的彼此相距 10 英里内成对的我方和敌方飞机，并了解何时发生这种情况"。传统的查询语言如 SQL 不能表达这样的查询。

• 索引 可以看到，数据库中的移动对象之数量可能是巨大的。因此从性能方面考虑，进行 MOD 查询我们希望避免检索在数据库中的每个移动对象的位置。换句话讲，要对位置属性建索引。直接使用空间索引的问题在于位置的连续变化意味着空间索引也必须连续修改，这同样是不能接受的方案。

• 不确定性/不精确性 一个移动对象的位置存在固有的不精确性，这是因为不论其采用何种方式修改数据库中对象的位置（即在数据库中存储的对象位置），这个数据库中的位置值总是与对象实际位置不同。此外，查询还存在两种不同类型的结果，即对象集合可能满足查询条件和对象集合必须满足这个查询条件。因此，针对查询必须提供不同的语义，并提供计算一个对象满足某个查询的概率的方法。

因此，需要将这样一个功能集成、构筑在现有 DBMS 的上部以支持移动对象数据库。下面将讨论可行的解决方案。

二、移动对象数据模型

在传统的 DBMS 系统中，数据是不变的除非被明确地修改。因此，为了表达在数据库中的移动对象（例如汽车），以及关于其位置的查询结果（例如，牌号为 rww80 的汽车与最近的旅馆的距离有多远?），小汽车的位置数据必须不停地进行修改，这是不可行的。要么位置修改过于频繁（这会严重影响系统的性能），要么

查询的结果过时了。因此需要采用能够处理动态属性的数据模型，动态属性是时间的连续函数。也就是说，对一个查询的回答不仅依据数据库的内容，而且要依据查询发生的时间，相反，一个对象的静态属性就是传统意义上的属性，即当数据库发生明确的变化，它才变化。

移动对象的动态属性是其位置属性，通常，一个动态属性 A 可以由三个子属性组成：A. updateValue，A. updatetime 和 A. function，其中 A. function 是时间 t 的函数，当 t＝0 时，A. function＝0。动态属性的值由时间决定，其定义如下：在时间 t＝A. updatetime，A 的值为 A. updateValue。时间为 t＝A. updatetime＋t0(t0 为正数)，A 的值变为 A updateValue＋A function (t0)。

在二维空间中，对于移动对象的位置属性 L 的建模可由二个动态属性 L. x 和 L. y 来表示。每一个都有其自己的修改值、函数(功能)以及更新时间来独立地表示对象的 x 和 y 坐标(所有的观点都可以延伸到三维空间中)。当对象的速度发生变化时，则要修改其位置数据。这对于在空间自由移动的对象(例如飞机)是一种直接的方法。但对于曲折前进的对象而言，它不是一种有效的方法，因为每次变向都调用 L. x. function 和 L. y. function。

为了解决这个问题，可以扩展动态属性将当前运动路线包括其中。其思想是：位置属性拥有 5 个动态子属性，即 L. route，L. x. updateValue，L. y. updataeValue，L. updatetime 和 L. speed。其中 L. route 是一条线空间对象，它描述了对象移动的线路；L. x. updateValue，L. y. updateValue 是 L. route 上的一个点的 x 和 y 坐标，是移动对象在时间 L. updatetime 的位置；L. updatetime 是上次位置修改的时间，L. speed 是一个线性函数，其形式为：f(t)＝b * t，由移动对象的速度决定，并通过它可以计算出从 L. updatetime 开始后经过时间 t 后当前位置与起始位置的距离。在时间为 L. updatetime＋t 时，位置为(x，y)，点(x，y)是在线路上距点(L. x. updateValue，L. y. updateValue)距离为 L. speed * t 的点。

三、移动查询语言

一个非时变的查询是与当前时刻相关的。例如，“检索当前在多边形 P 中的所有对象”。一个常规的查询语言如 SQL 或 OQL，在带时间谓词时就可以使用户对移动对象进行非时变的查询。但对于下面有时间变化的查询 Q：“检索所有进入多边形 P 的对象对(o，n)，并且要求在 o 和 n 之间的距离在 5 公里之内”。在 SQL 或 OQL 中表达这样一个时间性查询将非常繁琐。假定对于每个谓词 G，可采用 Begin_time(G)和 End_time(G)函数来提供满足 G 的一个时间区间的起始时间和终止时间。并假定用“NOW”来标识当前时间，则查询 Q 可以表示为：

```
RETRIEVE o,n
FROM Moving_objects
WHERE Begin_time(DIST(o,n)<=5)<=NOW
  AND End_time (DIST(o,n)<-5)>=
    Begin_time(INSIDE(o,P) A INSIDE(n,P))
```

其中，DIST(o，n)和 INSIDE(o，P)都是空间方法。DIST(o，n)返回 O 与 N 的距离，而 INSIDE(o，P)表明 o 是否在 P 之内。

移动查询语言应提供两个基本的时间性谓词，即 Until 和 NextTime。规则 f Until g 成立，当且仅当下列两种情况之一成立：要么 g 满足这个状态，要么存在一个将来的状态 g 是满足的，使得在此之后成立。规则 NextTime f 成立，当且仅当在 f 的变化规律中的下一个状态成立。因此在移动查询语言中，以上查询 Q 可以表述为：

```
RETRIEVE o,n
FROM moving_objects
WHERE DIST(o,n)<=5
 Until(INSIDE(n,P) A INSIDE(o,P))
```

查询语言还应该支持，诸如下列有界时间表达式：

· Eventually_within(c，g)表明规则 g 将在下一个 c 时间单元内成立。

· Eventually_after(c，g)表明 g 至少保持 c 个时间单。

· Always_for(c，g)表明 g 将在下一个 c 时间单元内总是成立的。

· (g Unitil_within(c，h))表明将存在一个实例，在 h 持有的 c 个时间单元内，g 可以连续保持。

一个移动查询可以有下列语法格式：

```
RETRIEVE (target_list)
WHREE (condition)
```

此处，(condition)由公式给定。例如，下面的查询用于检索在 3 个单位时间内进入多边形 P 中的对象 o，并且其属性值 PRICE<＝100。

```
(1) RETRIEVE o
    WHERE o. PRICE <= 100 A Eventually_within
    (3,o,P)
```

下面的查询用于检索在 3 个单位时间内进入多边形 P 中的对象 o，并且在 P 中停留 2 个单元时间。

```
(2) RETRIEVE o
    WHERE Eventually_within (3,INSIDE(o,P)
        A Always_for_INSIDE(2,o,P))
```

四、动态属性索引

本节我们讨论动态属性建立索引的问题，旨在能够进行涉及区间的查询，形如“检索当前在多边形 P 中的对象”或“检索在时间 t 时，其动态属性值在[ab…ae]范围内的所有对象”。直接使用空间索引存在的问

题是对象连续地改变位置，则空间索引也必须连续更改，这种方案是不可行的，可以采用下列方案来解决这个问题。索引问题可以分解为二个子问题，即移动对象的几何表示以及对几何表示的索引。

对于第一个问题，可以采用下列表示方法，即值-时间表示空间法（Value-Time Representation Space）和截距-斜率表示空间法（Intercept-Slope Representation Space）。

·值-时间表示空间法　描绘出所有函数，它们表达了动态属性随时间变化的方式。因此，这种方法表示的空间的构造是：x 轴表示时间，y 轴表示动态属性的值。一个对象被映射到一条轨迹上，这条轨迹描绘了对象的位置，它是时间的函数。

·截距-斜率表示空间法　对象 o 的位置是时间的函数，可以表示为 f(t)=a+v*t，a 称为截距，v 称为斜率，则表示空间的构造为：x 轴表示截距，y 轴表示斜率。因此，对象 o 就映射到这个空间中的点(o,v)，可以看到在以上的两种方法中，在空间中对象表示的修改当且仅当位置动态属性的一个子属性明确地被改动时才进行，因此第一种方法查询效率较高但修改效率较低，而在第二种方法中情形相反。

以上方法中的任一个都可以用于描述自由移动的对象。对于在路线上移动的对象找到一个有效的几何表示这个问题目前是一个公开问题。能有效地解决第二个问题的方法目前还在探讨之中。

五、不确定性管理

移动对象位置的不精确性是固有的，在移动对象位置数据库（即存放在数据库中的移动对象的位置数据）中，不管采用哪种修改策略，数据库位置不会总是与实际位置一样。这种不精确性与数据库建模、查询、索引都有不同的关系。

5.1　数据建模

为了对数据库位置的不精确性建模，首先要定义偏差的概念。通常，一个动态属性（在特定时间点 T）值的偏差是指在时间 T 的实际值与数据库值（即在数据库中的值）的差别。

对于不确定性建模的一个方法是提供偏差的边界。在任何时间点移动对象和 DBMS 都知道这个边界，并且当偏差达到边界时，移动对象就发送一个修改。因此，若边界为1公里。那么 DBMS 对一个查询："M 的当前位置是什么？"，其回答为："当前位置是(x,y)，半径为1公里的圆的面积"。对于沿着一个线路移动的对象，不确定性就是在点(x,y)前1公里和后1公里的一个区间。偏差的边界由另一个子属性 L.Uncertainty 给定。

5.2　查询语言

比如有一个查询 q="检索在多边形 P 内的对象"，由于数据库位置的不确定性，对这个查询存在两种不同类型的语义，即 MAY 和 MUST。在"MAY"语义下，回答是所有可能在 P 之内的对象的集合。在"MUST"语义下，回答是所有一定在 P 之内对象的集合。要处理这种不确定性问题，一种通用的方法是将概率与查询结果联系在一起。例如，查询 Q 的结果可以是对象1在 P 中的概率为0.4，对象2在 P 中的概率是1。

5.3　索引

既然赋予了查询语义，索引也需要进行扩展以有效地处理以下查询：Q1="检索在时间 T一定在多边形 P 内的对象"或 Q2="检索在时间 T 可能在 P 内的对象"。比如可以通过对值-时间表示空间进行扩展（参见第4节）来处理不确定性。构造一个平面，称为 o-平面来表示移动对象 o 的位置属性。这个 o-平面是 o 的不确定区间的集合，对于每个时间单元 t≥0，存在一个不确定区间。因此，不用一条线（或一条轨迹）来表示，而采用一个平面来表示一个对象。换句话来说，在时间 t，位置属性的值是一个区间而非一个点。一个区间查询如前所述仍然是一个线段。对上面查询 Q1的结果就是一个对象集合。在时间 T，这些对象的不确定区间完全在 P 之内，Q2的查询结果是：在时间 t 不确定区间与 P 相交的对象集合。

5.4　不确定性和通讯的折衷

虽然，数据库的位置与移动对象实际位置有偏差，但采用较高频率的修改能够减少偏差。显然，在通讯和不精确性之间存在折衷的考虑。在5.1节中所提出的模型上，不精确度可由偏差的边界获得，本小节要解决的主要问题是如何定义由 L uncertainty 标识的边界。可以采用基于开销的方法来解决这个问题。移动对象在移动过程中的开销由以下三部分组成：

·偏差开销　首先可看到偏差的损失依赖于偏差的大小和它存在时间的长短。偏差的大小会明显地影响决策。对于偏差存在时间的长短，现假定有一个查询，它每个单位时间都检索移动对象的位置。那么，若偏差存在二个单位时间则它的损失将是存在一个单位时间的两倍；其原因是有两个查询（而不是一个查询）要承担偏差损失，对于一个移动对象 m，在两个时间点 t_1，t_2的偏差损失由偏差函数给定，记为 $COST_d(t_1,t_2)$，即

$$COST_d(t_1,t_2)=\int_{t_1}^{t_2} d(t)dt \qquad (1)$$

其中 d(t)表示偏差，是时间的函数。

·修改损失　记为 C_1，是一个非负数，表示位置修

改消息从移动对象传到数据库的损失。一个移动对象的修改损失不同于另一个移动对象的，并且可能在移动对象移动过程中发生变化（比如由于可用带宽发生变化而变化）。

·不确定性损失　当回答一个查询时，不确定性越高传送回来的信息越少。可以看出，象偏差一样，不确定性损失依赖其大小和其存在时间的长短。我们给出形式上的描述，对于一个移动对象 M 在时间点 T_1，T_2 之间的不确定性损失由下面不确定性损失函数给定：

$$COST_u(t_1,t_2)=\int_{t_1}^{t_2}C_2u(t)dt \qquad (2)$$

其中，u(t)为 m 的子属性 L. Uncertainty 的值，是时间的函数。C_2是不确定性的单位损失。

那么在半开区间$[t_1,t_2)$移动带来的整个损失为：

$$COST_I[t_1,t_2)=C_1+COST_d[t_1,t_2)+COST_u[t_1,t_2) \qquad (3)$$

从而移动对象在整个移动过程中的全部损失为：

$$COST_I=COST_d[0,t_1)+COST_u[0,t_1)+\sum_{i=1}^{k}COST[t_i,t_{i+1}) \qquad (4)$$

其中 $t_1,t_2,\cdots,t_k$ 是由 m 发出的修改消息的时间点，0 为起始点；t_{k+1}为旅行的终点。

一个移动对象的偏差边界（或阈值），即 L. Uncertainty 子属性，决定了适合采用何种位置修改策略。可以采用的方法有：

(1)在旅行开始，移动对象 m 向 DBMS 发送一个不确定性值的大小，并存储在 L. Uncertainty 中，在整个移动过程中，值固定不变。一旦偏差超过 L. Uncertainty 时，对象 m 就修改数据库中的当前位置和速度。

(2)移动对象为每个修改提供了一个新的不确定性值 th，th 是利用基于损失的方法来计算得到的。为了计算新的不确定度，移动对象 m 要预测偏差。不确定度对于不同的修改策略是不同的。经过分析表明最优的不确定度为$\sqrt{2aC_1/(2C_2+1)}$。其中，a 是偏差的近似斜率。C_1是修改损失，C_2是不确定性的单位损失。

在这两种策略中，第一种对于所有位置修改是固定值，而在每对连续修改之间不确定度是固定的，第二种在每对之间是变化的。

六、系统实现

基于以上讨论，在现有 DBMS 上实现这样一个系统包括4个子系统，即查询处理子系统、移动对象子系统、策略模拟子系统和策略修改子系统。

·查询处理子系统　实现支持移动对象的数据模型和移动查询语言。查询 GUI 用于进入移动查询和查看查询结果。

·移动对象子系统　在移动对象上建立局部子系统，使用数据库触发器实现修改策略，对象的当前位置保存在局部数据库中，并以固定频率（如1秒）进行修改。局部数据库由支持触发器的 DBMS 管理。

·策略修改子系统　实现修改策略和相应评估算法，它能定义新的策略同时并不影响整个系统的其他部分。

·策略评估子系统　根据信息全部损失，即修改、不确定性、偏差的总和来评估不同的修改策略，它以速度曲线作为输入，这个曲线是时间的函数。用户可以通过在屏幕上划一条线表示速度曲线作为输入。

对于每个速度曲线，修改策略，修改损失 C_1和不确定性单位损失 C_2，策略评估子系统执行一次模拟运行。这次运行要计算在曲线上的信息损失。那么对于每个策略，策略评估子系统对这个速度曲线的信息损失进行平均，并描绘出这个平均值随着修改损失 C_1的函数变化趋势。

每次模拟执行中，一个速度曲线是实际速度的序列 S。每个时间有一个速度值，策略评估子系统模拟特定修改策略工作的移动计算机。其做法如下：

对于每个时间单元存在一个不确定值 th，以及数据库中对象的速度和实际速度。实际速度由速度曲线给定。在某个特定时间点 t 的偏差是实际速度（整数、时间的函数）与数据库速度之间的差。偏差序列记为 T。在每个单位时间有一个值。不确定阈值由序列 Q 标记，那么策略评估子系统产生一个修改记录，包括：当前时间，当前位置，当前速度，下一个不确定值（不确定值的计算参见前面所述）；在时间 T，策略评估计算偏差的所有损失，记为 C_1，使用 U 计算偏差的所有损失记为 C_2，利用 Q 计算不确定性的所有损失记为 C_3。在速度曲线上所有信息损失为 $C_1+C_2+C_3$。

在每个位置修改，策略评估子系统激活策略修改子系统来计算新的偏差。策略评估子系统保存速度文档和模拟结果。此外，策略评估子系统允许在模拟运行期间修改参数。

结束语　移动对象数据库是数据库系统的一个发展方向。本文提出的解决方案及原型系统是不依赖于任何特定的 DBMS，因此具有较大的通用性。文中所提出的思想对相关研究领域也是有一定参考意义的。

参 考 文 献

[1] Alonso R, Korth H F. Database System Issue in Nomadic Computing. In: Proc. of the 1993 ACM SIGMOD Intl. Conf. On Management of Data, Washington, DC, 1993

[2] Wolfson O, Jiang L, et al. Updating and probilistic querying of motion database, submitted for publication, 1998

[3] Sistla A P, Wolfson O, et al. Modeling and Quering Moving Objects. In: The Proc. of the IEEE Intl. Conf. on Data Engineering, 1997