第 26 卷 第 5 期 计算机工程与设计 2005 年 5 月
Vol. 26 No. 5 Computer Engineering and Design May. 2005

一种增量更新 FreeCube 的方法

翁 伟[1]， 薛永生[1]， 王劲波[2]
(1. 厦门大学 计算机科学系，福建 厦门 3610051；2. 厦门大学 计划统计系，福建 厦门 361005)

摘 要：数据立方体是联机分析处理的一个重要应用。如何对数据立方体（CUBE）进行更新目前研究相对较少。给出了 CUBE 的一种新颖的存储结构——FreeCube 的定义，该结构大大降低了 CUBE 存储的空间，分析了它的相关性质，提出了增量更新 FreeCube 的理论，并给出了具体的算法，以实例说明了该算法的正确性，总结了下一步的工作方向。
关键词：数据立方体；联机分析处理；FreeCube
中图法分类号：TP311.13 **文献标识码**：A **文章编号**：1000-7024 (2005) 05-1199-04

Method of incrementing updating FreeCube

WENG Wei[1], XUE Yong-sheng[1], WANG Jin-bo[2]
(1. Department of Computer Science, Xiamen University, Xiamen 361005, China; 2. Department of Planning Stat, Xiamen University, Xiamen 361005, China)

Abstract: Data cube is an important application in OLAP system. The study on how to update data cube lags behind at present. A kind of the novel structure of CUBE——Free is described that can reduce the size of CUBE quickly. And then the characteristics of the constrictive freecube is analyzed. The theory of incrementing updating freecube is proposed, and algorithm for it and then a instance to indicate the correctness of the algorithm is provided. At last, the directions in the future is summarized.
Key words: data cube; OLAP; FreeCube

1 引 言

联机分析处理 (On Line Analysis Processing，简称 OLAP)是数据仓库的一个重要的应用。OLAP 应用是面向主题的，侧重于历史数据的汇总、归并和分析。OLAP 的核心操作是对关系数据进行大量的分组 (GROUP BY) 求和。为此，Dr. Jim Gray 等人在 ICDE '96 大会上首先提出了数据立方计算[1]的问题。以数据立方算子 CUBE BY 是通常的 GROUP BY 操作的扩展。例如，考虑一个基表 SALES (date，product，customer，amount)，数据立方查询：

```
SELECT date，product，customer，SUM(amount)
FROM SALES
CUBE BY date，product，customer
```

将对 CUBE BY 的属性 date、product 和 customer 的所有组合对应的 8 个 GROUP BY 上进行求和聚集计算，即 (date，product，customer)、(date，product)、(date，customer)、(product，customer)、(date)、(product)、(customer)和 ALL。

为了尽量快速回答用户的聚集查询，一般把数据立方体实物化，即把在多维数据立方体上的切片或切快的聚集结果保存在物理存储上。许多学者在多维数据库的基础上对此进行了很有成效的工作。研究工作一般围绕 3 个问题：①如何快速算出 CUBE[2]；②计算时如何尽量少占内存；③如何压缩多维数据立方体以及 CUBE[3]。以上 3 个方面的研究现在已经做得非常深入，理论成果非常多。但是对于建立起来的 CUBE 如何进行更新，这方面的理论研究相对少得多。

数据仓库不断地从业务处理系统或企业外部获得数据，然后在进行抽取、清洗和转换等操作后保存下来供联机分析处理所用。因此数据仓库一个不断增大的系统，这就意味着数据仓库中更抽象层次的聚集数据——CUBE 也必须得到更新。对于如何更新已经建立起来的 CUBE，一般有两种方法：一种是重新计算 CUBE，这种方法的缺点是显而易见的，一般代价高，若数据仓库的数据丢失或人为清除了一些历史数据，则这种方法就不可能进行了；第 2 种方法就是增量更新。这种方法的优点很多，它不需要以前的历史数据，仅仅从新计算新加进数据仓库的数据即可。在需要更新的数据比例较小时，或在精度要求不高以及重新计算代价较大等环境中 (如数据库是分布式的)，一般进行增量更新。

文献[4]结合 free-set 的概念，给出了基于 BUC 算法[5]的一种高效率减少 CUBE 体积的方法，即算法 FreeCube。该方法一方面计算速度较快，另一方面却不是压缩的，它是利用元组

收稿日期：2004-05-08。 基金项目：福建省自然科学基金项目 (A0310008)；福建省高新技术研究开放计划重点基金项目 (2003H043)。
作者简介：翁伟 (1979-)，男，湖南衡阳人，硕士生，研究方向为分布式数据库、数据仓库和数据挖掘等； 薛永生 (1946-)，男，福建福清人，教授，研究方向为数据库理论与应用、分布式数据库、数据仓库、数据挖掘、网络技术和计算机应用技术等； 王劲波 (1978-)，男，湖南益阳人，助教，硕士，研究方向为分布式数据库、实时数据库、Web 数据库和数据挖掘。

间的联系去除冗余信息得到的，因此在回答查询时也不需要解压缩，论文中提供了构造性证明说明可以回答查询即完整性。但是关于 FreeCube 的更新，作者却没有看到相关的研究成果，因此有必要进行这方面的研究。本文在给出 FreeCube 相关概念和性质之后，提出了一种增量更新 FreeCube 的方法。

2 Free Cube 的定义和性质

对于一个有 n 维属性 $(D_1, D_2, \cdots, D_n)$ 和一个度量属性 M 的基本关系表 R，一个 Cube 等价于在 n 维属性集合的不同子集上的 2^n 个 GROUP BY 操作。每个 GROUP BY 对应一个 cuboid[6]。包含 k 个属性 $(0 \le k \le n)$ 的 GROUP BY 对应的 cuboid 称为 k 维 cuboid。通过对每一个维扩展一个维值 ALL，使得 data cube 的结果能用基本关系表 R 的一致模式表示。Data Cube 中的元组称为 Cube 元组。

定义 1 （蕴涵规则） 在一个 n 维的基本关系表 R 中，$d_1, d_2, \cdots d_i, d_m$ 分别是不同维上的某个维值（均不为度量属性值），如果基本关系表中出现 $(d_1, d_2, \cdots d_i)$ 的所有元组在 d_m 对应的维上值都为 d_m，则称 $(d_1, d_2, \cdots d_i)$ 蕴涵 d_m，记为 $R(d_1, d_2, \cdots d_i) \Rightarrow R(d_m)$。

不妨称该蕴涵规则的前一部分 $\{d_1, d_2, \cdots d_i\}$ 称为前件集，后一部分 d_m 称为后件。同一前件可能会对应许多后件，则这些后件的集合称为后件集。关系表 R 中 $(d_1, d_2, \cdots d_i)$ 的后件集记为 $\mathrm{SUFFIX}_R(d_1, d_2, \cdots d_i)$。

表 1 基本关系表 R

T	S	P	M
T1	S1	P1	10
T1	S1	P2	20
T2	S1	P1	40

例如：在表 1 中，出现 $T1$ 的元组就出现了 $S1$，所以根据定义有规则 $T1 \Rightarrow S1$；但是却没有规则 $R(T1) \Rightarrow R(P1)$，因为出现 $T1$ 的元组对应 P 那一维的值有些是 $P1$，有些却是 $P2$。

引理 1 若 $R(d_1, d_2, \cdots d_i) \Rightarrow R(d_m)$，且 $d_1', d_2', \cdots d_j'$ 是与 $d_1, d_2, \cdots d_i, d_m$ 不同维上的值，则有 $R(d_1, d_2, \cdots d_i, d_1', d_2', \cdots d_j') \Rightarrow R(d_m)$。

例如，如表 1 所示，因为有规则 R (T1) ⇒R (S1)，故也有 R (T1，P1) ⇒ R (d$_m$)。

引理 2 若 $R(d_1, d_2, \cdots d_i) \Rightarrow R(d_m)$，给定聚集函数（例如求和函数 SUM）$f$，$f_R$ 表示在关系表 R 上进行聚集函数 f 的操作，则 $f_R(d_1, d_2, \cdots d_i, d_m) = f_R(d_1, d_2, \cdots d_i)$，其中 $f_R(d_1, d_2, \cdots d_i)$ 表示对含有 $d_1, d_2, \cdots d_i$ 的所有元组的聚集，$f_R(d_1, d_2, \cdots d_i, d_m)$ 表示对既含有 $d_1, d_2, \cdots d_i$ 又含有 d_m 的所有元组的聚集。

定义 2 （free-set） 在一个 n 维的基本关系表 R 中，$d_1, d_2, \cdots d_i$ 分别是不同维上的某个维值（均不为度量属性值），如果对任何不同于 $d_1, d_2, \cdots d_i$ 所在维的一个维值 d_m（不为度量属性值），都不能得到 $R(d_1, d_2, \cdots d_i) \Rightarrow R(d_m)$，那么称 $(d_1, d_2, \cdots d_i)$ 是一个 free-set。如果 $i=n$，则显然 $(d_1, d_2, \cdots d_i)$ 是一个 free-set。若一个 free-set 的任何子集都不是 free-set，则称该 free-set 为最小 free-set。

例如，表 1 中 $(S1)$ 是 free-set，$(T1, S1)$ 也是 free-set。

引理 3 若 $(d_1, d_2, \cdots d_i)$ 不是一个 free-set，则必是某一蕴涵规则的前件集。

证明： 用反证法易证。

引理 4 若 $(d_1, d_2, \cdots d_i)$ 不是一个 free-set，则有某一个不同于 $d_1, d_2, \cdots d_i$ 所在维的一个维值 d_m，有规则 $R(d_1, d_2, \cdots d_i) \Rightarrow R(d_m)$，那么对于 $d_1', d_2', \cdots d_j'$，若 $d_1', d_2', \cdots d_j'$ 所对应的维与 $d_1, d_2, \cdots d_i, d_m$ 所对应的维都不同，则有 $(d_1, d_2, \cdots d_i, d_1', d_2', \cdots d_j')$ 也不是 free-set。该性质可由定义 2 和引理 2 得到。

定义 3 （Free Cube） 利用基本关系表得到所有 free-set，由这些 free-set 产生的 cube 元组构成了一个 Free Cube。

表 2 就是用算法 FreeCube 对表 1 进行处理生成的 Free Cube。其中‘ * ’表示 ALL。

表 2 Free Cube (R)

Cuboid	T	S	P	M
TSP	T1	S1	P1	10
—	T1	S1	P2	20
—	T2	S1	P1	40
TS	T1	S1	*	30
SP	*	S1	P1	50
S	*	S1	*	70

下面为了更进一步讨论 Free Cube 的性质，特做假设如下：R_1 和 R_2 都是基本关系表，它们的关系模式相同，即它们的属性和度量属性相同。设由 R_1 生成的 FreeCube 记为 FreeCube (R_1)，由 R_2 生成的 FreeCube 记为 FreeCube (R_2)。R_1 与 R_2 合并成表记为 R，由 R 生成的 FreeCube 记为 FreeCube (R)。

引理 5 若在基本关系表 R_2 中，$(d_1, d_2, \cdots d_i)$ 是 free-set，$X = \{d_1, d_2, \cdots d_j\}$ $(1 \le j < i)$，$Y = \{d_1, d_2, \cdots d_k\}$ $(k \ge i)$，为了叙述方便，不妨称 X 为集合 $\{d_1, d_2, \cdots d_j\}$ 的前缀子集，Y 是集合 $\{d_1, d_2, \cdots d_k\}$ 的后缀扩充集。若 X、Y 是 R_1 的 free-set，并且在关系表 R_2 中，包含 X 的所有后缀扩充集中，长度最小的 free-set 为 X'，在关系表 R_1 中，包含 $\{d_1, d_2, \cdots d_i\}$ 的最少 free-set 是 Y'，则在 Free-Cube (R) 中，有 $f_R(X) = f_{R1}(X) + f_{R2}(X')$，$f_R(d_1, d_2, \cdots d_i) = f_{R2}(d_1, d_2, \cdots d_i) + f_{R1}(Y')$。$X'$ 和 Y' 均称为最小后缀扩充 free-set。

证明： 根据假设，因为 R 包含 R_1 和 R_2，所以 R_1 和 R_2 中的 free-set 同样也是 R 的 free-set，因此 $(d_1, d_2, \cdots d_i)$、X 和 Y 都是 R 的 free-set。因为 $f_R(X) = f_{R1}(X) + f_{R2}(X)$，有根据引理 2，有 $f_{R2}(X) = f_{R2}(X')$，所以有 $f_R(X) = f_{R1}(X) + f_{R2}(X')$。同理有 $f_R(d_1, d_2, \cdots d_i) = f_{R2}(d_1, d_2, \cdots d_i) + f_{R1}(Y)$。

引理 6 若在基本关系表 R_1 和 R_2 中，$(d_1, d_2, \cdots d_i)$ 都不是 free-set，但在 R 中，$(d_1, d_2, \cdots d_i)$ 是 free-set，则 $\mathrm{SUFFIX}_{R1}(d_1, d_2, \cdots d_i) \cap \mathrm{SUFFIX}_{R2}(d_1, d_2, \cdots d_i)$ 为空集。

证明： 在基本关系表 R_1 中，$(d_1, d_2, \cdots d_i)$ 不是 free-set，所以 $(d_1, d_2, \cdots d_i)$ 是某一条蕴涵规则的前件，其后件为 $\mathrm{SUFFIX}_{R1}(d_1, d_2, \cdots d_i)$，同理也有 $\mathrm{SUFFIX}_{R2}(d_1, d_2, \cdots d_i)$ 成立。若 $\mathrm{SUFFIX}_{R1}(d_1, d_2, \cdots d_i) \cap \mathrm{SUFFIX}_{R2}(d_1, d_2, \cdots d_i)$ 不为空集，不妨设 $d_m \in \mathrm{SUFFIX}_{R1}(d_1, d_2, \cdots d_i) \cap \mathrm{SUFFIX}_{R2}(d_1, d_2, \cdots d_i)$，则有 $R_1(d_1, d_2, \cdots d_i) \Rightarrow R_1(d_m)$，$R_2(d_1, d_2, \cdots d_i) \Rightarrow R_2(d_m)$，且 R 由 R_1 与 R_2 组成，所以也有 $R(d_1, d_2, \cdots d_i) \Rightarrow R(d_m)$，这与 $(d_1, d_2, \cdots d_i)$ 在 R 中是 free-set 矛盾，所以原命题成立。

上面的定义和引理是算法讨论的理论基础。

3 Free Cube 的增量更新算法

R_1和R_2都是基本关系表，它们的关系模式相同，即它们的属性和度量属性相同。在数据仓库中，已经生成了 FreeCube(R_1)，下面讨论如何把基本关系表R_2增量更新到 FreeCube(R_1)中去。在此采用的办法是借助算法 FreeCube，在生成R_2的 free-set 的同时，把该 free-set 在R_2上的聚集元组添加到 FreeCube(R_1)中，这不是简单地在 FreeCube(R_1)中新增一条聚集元组，而要分两种情况讨论，讨论建立在 BUC 算法之上：

(1) 如果在R_2中生成了一条 free-set($d_1,d_2,\cdots\cdots d_n$)，则根据引理 5，在 FreeCube(R_1)中找出集合$\{d_1,d_2,\cdots\cdots d_n\}$的一个前缀子集$X$，如果$X$是$R_1$中的 free-set，则求出$X$在$R_2$中的最小后缀扩充 free-set X'。由于 BUC 算法是一种自底向上的计算聚集的方法，所以此时$f_{R2}(X')$早已计算出来。然后若X上没有已经更新的标志，则先计算$f_{R1}(X)=f_{R1}(X)+f_{R2}(X')$，把$f_{R1}(X)$记录到 FreeCube($R_1$)中去后，再把 FreeCube($R_1$)中该聚集元组作标志表示已经更新好，下次不再更新；然后再生成集合$\{d_1,d_2,\cdots\cdots d_n\}$在$R_1$中的最小后缀扩充 free-set Y'，计算$f_{R1}(d_1,d_2,\cdots\cdots d_n)=f_{R2}(d_1,d_2,\cdots\cdots d_n)+f_{R1}(Y')$。注意这里作一个标志和计算顺序特别重要。若不作标志，会导致重复计算；若计算顺序不对，计算结果不会正确。若集合$\{d_1,d_2,\cdots\cdots d_n\}$在 FreeCube($R_1$)即没有前缀子集，又没有最小后缀扩充 free-set Y′，说明集合$\{d_1,d_2,\cdots\cdots d_n\}$中有以前未出现过的值，则直接计算$f_{R2}(d_1,d_2,\cdots\cdots d_n)$加到 FreeCube（$R_1$）中去。设实现该功能的过程为 UPDATE-ONE，参数为一个 free-set。

(2) 如果在R_2中生成了一条蕴涵规则，不妨设该蕴涵规则为$(d_1,d_2,\cdots\cdots d_n)\Rightarrow R(d_m)$，这种情况比较复杂。若此时在 Free-Cube(R_1)有 free-set($d_1,d_2,\cdots\cdots d_n$)，则根据第 1 种情况在后面会得到更新。若没有 free-set($d_1,d_2,\cdots\cdots d_n$)，则必须考虑是否会产生新的 free-set($d_1,d_2,\cdots\cdots d_n$)。判断的办法是先在 FreeCube(R_1)找出$f_{R1}(d_1,d_2,\cdots\cdots d_n)$记录的属性值集合，根据引理 4，可以得到 SUFFIX $_{R1}(d_1,d_2,\cdots\cdots d_n)$，再根据引理 6，可以判断$(d_1,d_2,\cdots\cdots d_n)$是否是新生成的 free-set。若是，则在 FreeCube(R_1)增加一条新的聚集元组。设实现该功能的过程为 UPDA-TETWO，参数为一条蕴涵规则。

下面用伪码描述增量更新 Free Cube 算法。

```
Algorithm add_FreeCube(input_R2, dim_R2)
1 :if Count(input_R2) = =1 then  //如果只有一条元组
2 :UPDATEONE(input_R2)
3 :return;
4 :end if
5 :isFreeSet=true;
6 :for (d=0 ;d<numDims ;d++) do
7 :if (outputRec.dim [d]!=ALL) then //表示该属性为划分属性
8 :addprefix(preSet) ;//在前件集合preSet里增加一个元素
9 :continue;
10 :else
11 :if No.d dimension of input has unique value then
12 :ifFreeSet=false;
13 :addsuffix(sufSet) ;// 在后件集合sufSet里增加一个元素
14 :end if
15 :end if
16 :end for
17 :if (isFreeSet)
18 :then
19 :Aggregate(inputR2,collectRec) //进行聚集计算，形成聚集元组 collectRec
20 :UPDATEONE(preSet)
21 :else//preSet不是free-set，即是某蕴涵规则的前件集合，如引理 3 所说明
22 :UPDATETWO(preset,sufSet)
23 :end if
24 :for (d=dim ;d<numDims ;d++) do
25 :C=cardinality [d];//求出第 d 维集合的势，即在 d 维上可划分为 C 块
26 :Partition (input,d,C,dataCount [d]);
27 :k=0;
28 :for(I=0 ;I<C ;I++) do
29 :c=dataCount [d][I] ;//第 I 快的元组个数
30 :FreeCube(input[k…k+c],d+1);
31 :k+=c;
32 :end for
33 :end for
```

在算法中把不作为划分用的属性称做非划分属性，用“ALL”标志。算法首先有一个优化措施(1~4 行)，在输入只有一个元组的情况下，若存在非分区属性，则分区属性值的集合不可能是 free-set，只有整个元组所有的属性值之集合才是 free-set，所有此时直接计算该 free-set 的聚集，同时返回调用，不再进行第 24 行的分区递归，因为一条元组再分区也只有一个分区，即该元组。第 5~16 行判断划分属性是否是 free-set。划分属性值都保存在 preset 里。若划分属性的集合 preset 不是 free-set，则就是某一蕴涵规则的前件集合，相应的后件集合在 sufset 里。第 11 行判断 preset 是否是 free-set，方法就是看该划分中所有的元组在某一非划分属性的值上是否相同，若相同则不是 free-set。第 17~21 行进行 free-set 的聚集计算并添加到 Free-Cube(R_1)中去。第 20 行的函数 UPDATEONE() 和第 22 行的函数 UPDATETWO() 在前文已详细讨论。第 24~33 行是递归的对非划分属性继续进行划分的过程。

4 实例分析

我们在数据库中保存表 2 的数据模拟一个数据仓库中的 FreeCube，然后对于如下所示的表 3 中基本关系表 R ′的数据，采用上面的算法将表 3 的数据加到数据仓库(表 2)之中。

算法的运行结果如表 4 所示。下面解释一下表 4 某些记录的生成过程。

算法在表 3 上运行时，就会发现一条 free-set($T1,P3$),但是集合$\{T1,P3\}$在表 2 的 Free Cube(R) 中既不存在前缀子集，又不存在最小后缀扩充 free-set，所以直接计算 $f_{R'}(T1,P3)$加入新的

Free Cube 之中。同理，表 3 中有 free-set (T1,S1,P3), 集合{T1,S1,P3}在表 2 中有前缀子集{T1,S1}，而集合{T1,S1}在表 3 中的最少扩充 free-set 是(T1,S1,P3), 则表 4 新的 Free Cube (R) Cuboid 列值为“TS”的记录值为原来表 2 中 Free Cube 该列的值 30 加上表 3 中相应的记录值 10 的结果。其余的记录产生的过程类似。

同时我们把表 1 与表 3 结合成一个表，运行生成 Free Cube 的程序得到的结果同表 4 是一样的，这表明本文提出的增量更新算法是有效的。

通过该实例可以发现，新生成的 Free Cube 绝不是一个简单的相加的过程。同时，尽管数据仓库中只保留 Free Cube, 对于新加入的基础数据，也可以很好地产生新的 Free Cube。但是如果数据仓库中的 Free Cube 是有缺失的，对产生的结果会有影响。缺失越多，负面影响越大。

表 3　基本关系表 R'

T	S	P	M
T1	S2	P3	30
T1	S1	P3	10

表 4　新的 Free Cube (R)

Cuboid	T	S	P	M
TSP	T1	S1	P1	10
—	T1	S1	P2	20
—	T2	S1	P1	40
—	T1	S2	P3	30
—	T1	S1	P3	10
TS	T1	S1	*	40
TP	T1	*	P3	40
SP	*	S1	P1	50
T	T1	*	*	70
S	*	S1	*	80

5　结束语

在数据仓库中历史数据丢失等情况下，对 CUBE 进行增量计算是不可避免的。本文对 CUBE 的一种缩少体积的存储结构——FreeCube 进行分析并提出相关性质之后，给出了增量更新 FreeCube 的算法，具有一定的理论价值和实用价值。

关于 FreeCube 还有许多地方值得研究。下一步的工作主要集中在以下几个方面：①如何通过 FreeCube 重新恢复历史数据，比如在分布式环境中，个别场地上需要历史数据，而从中央数据库传输大量数据代价高，或是历史数据丢失，就有这种需求；②设定支持度和致信度，通过研究 FreeCube 如何识别出其中的关联规则；③如何定义一种高效的 FreeCube 存储结构以提高 FreeCube 响应用户查询的速度。

参考文献:

[1] Gray J, Bosworth A, Layman A, et al. Datacube:A relational aggrefation operator generalizing group-by,cross-tab and sub-totals[C]. Louisiana, Washington:Proc ICDE New Orleans, DC: IEEE Computer Society, 1996. 152-159.

[2] 向隆才, 冯玉才, 肖凌. 一种提高 BUC 算法效率的有效方法[C]. 计算机科学, 2003, 30(增刊):233-237.

[3] FENG Yu, WANG Shan. Commpressed data cube for approximate OLAP query processing[J]. Comput Sci &Technol, 2002, 17(5):625-635.

[4] 孙延凡, 陈红, 王珊. FreeCube:有效减少 Data Cube 体积[J].计算机科学, 2003, 30(增刊):241-245.

[5] Beyer K, Ramakrishnan R. Bottom-Up computation of sparse and iceberg CUBEs[C]. Proc ACM SIGMOD Philadelphia, Pennsylvania, USA, New York: ACM press, 1999. 359-370.

[6] Harinarayan V, Rajaraman A, UIIman J D. Implementing data cubes efficiently[C]. ACM-SIGMOD, 1996. 205-216.

(上接第 1198 页)

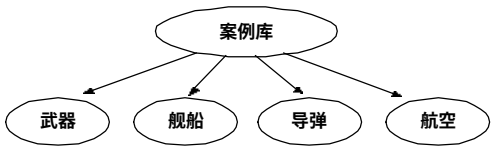

图 5　案例库的索引结构

名称作为案例库的索引文件驻留在内存中，将案例库放在硬盘上。在用户输入案例名称后，推理机根据该名称找到索引文件，然后可以根据名称索引进行模板检索，迅速地到案例库中搜索到具有相同名称的所有案例，从而提高系统检索效率。

4　结　论

本文结合 RCM 分析案例的主要构成特点，对案例库的组织结构和索引机制进行了研究，给出了面向对象、基于关系数据库的案例库组织形式。该研究具有积极的意义，运用该案例库构建技术初步实现了基于案例的 RCM 分析系统案例库的开发。实践证明，通过建立基于案例的 RCM 分析系统的案例库，可以简化和加速 RCM 分析过程，使 RCM 分析过程拥有了类似人类思维和智慧的能力，使基于案例的 RCM 分析系统达到了优化分析结果、减轻分析者脑力劳动、提高分析效率的目的。

参考文献:

[1] 贾希胜, 程中华. 以可靠性为中心的发展动态[J]. 军械工程学院学报, 2002,3(3):32-36.

[2] 童俯, 沈一栋. 知识工程[M]. 北京:科学出版社, 1992.

[3] 孙林夫. 工程智能化 CAD 系统设计[J].计算机应用, 1998,18(5):4-6.

[4] 康雪峰, 周洪玉. 基于数据库技术的面向对象知识表示[J]. 哈尔滨理工大学学报, 2001,6(3):1-3.

[5] 王永庆. 人工智能原理与方法[M].西安:西安交通大学出版社, 1998.

[6] 史忠植. 高级人工智能[M]. 北京:科学出版社, 1998.