在不同 DBMS 上的数据库转换*

唐世渭　杨冬青　裴　芳

(北京大学计算机科学系)

DATABASE CONVERSION BETWEEN DIFFERENT DBMSs

Tang Shiwei, Yang Dongqing, and Pei Fang

(*Department of Computer Science and Technology, Beijing University*)

ABSTRACT

In this paper, we introduce the concept and several methods of database conversion. Then we present a database conversion system, which converts information system from IMS to ORACLE. The system consists of three parts: schema conversion, data conversion, and application program conversion.

摘　要

本文介绍了数据库转换的概念和方法，并介绍了一个数据库转换系统，该系统将数据库及应用程序从层次模型的 *IMS* 系统转换到关系模型的 *ORACLE* 系统。转换系统包括三个部分：模式转换、数据转换、应用程序转换。

§1. 引　言

随着数据库技术的不断发展，数据库的应用范围越来越广泛。由于一些应用领域的需要，要求我们研究不同 DBMS 之间的相似与不同，并实现其上的数据模式、数据、数据操作之间的转换。

例如：现在已有很多建立在 DBMS 上的成功的信息系统，用户可能希望在具有较强数据管理功能和较好的用户界面的新的 DBMS 上建立具有同样功能的信息系统。如果由人工来完成这项工作，将花费较多的人力、物力，不仅代价很大，而且不易保证新旧信息

* 1989年4月15日收到，1989年7月5日定稿。

系统的一致性，因此，应建立数据库的转换系统，自动地将原来的信息系统转换为新的DBMS上的信息系统。

再如：在计算机网络中，各结点使用的DBMS不同的情况下，为实现各结点之间的数据共享，会希望在网络中有一个统一的数据库界面，或在每个结点上以自己的DBMS界面存取其它结点上的不同DBMS支持下的数据。这就需要每个结点的计算机能够将自己的数据模式转换为其它DBMS的数据模式，并将其它DBMS上的数据操作转换为自己数据库上的数据操作，使网络中的其它计算机能使用自己数据库中的数据。

此外，由于用户要求的改变，DBMS上信息系统的数据模式随之发生变化，在模式改变后，要将数据库中的数据按新模式加工后重新装入数据库，同时，改变原有的应用程序，以适应模式的变化。这些都要求我们研究数据库转换技术。

§2. 数据库转换概述

本文所说的数据库转换，是找出不同的DBMS在模式、数据、数据操作之间的对应关系，并按照这种对应关系，将原DBMS上信息系统的模式、数据、应用程序转换为目标DBMS上的模式、数据、应用程序，构成目标信息系统。

1. 模式转换

模式转换是数据库转换的基础。

数据库的数据模式由两部分组成:

(1)结构：是对实体数据和实体之间联系数据的抽象。

(2)约束：是数据上的逻辑限制，是实体数据或实体之间联系数据的一个或为真或为假的性质。

约束有以下两类:

* 内在的约束：这种约束是由结构决定的，是结构的有机组成部分。例如：关系数据库中元组不能重复。

* 显式的约束：通过显式说明对数据所加的限制。例如：说明年龄数据项大于零。

模式的转换要同时考虑结构和约束的转换，才能保证目标模式与源模式的一致性。

例如：网状数据库的数据模式由记录和系组成，记录是实体数据集，系反映了记录之间的联系，在系中说明属籍类别，就是在联系数据上说明的显式约束。

网状模式到关系模式的转换方法是:

(1)为网状模式中的每个记录Si，建立一个关系Ri，Si中的所有数据项，在Ri中都有相应的属性，如果Si有码，则Ri的码等于Si的码，否则，Si的DBkey作为Ri的码。

(2)如果Si是系中的主记录型，Sj是属记录型，则Ri的码作为Rj的外码，并根据系中说明的属籍类别建立外码约束。

这个转换方法的第一步保证了关系模式中保留了网状模式中的所有数据。第二步在关系模式中用外码的方式实现了网状模式中的系，并通过外码约束实现属籍类别。

2. 数据转换

模式转换完成后，要按照源模式和目标模式的对应关系将源数据库中的数据转换为符合目标模式的数据.

数据转换的基本过程分三步:

(1)读出源数据库中的数据.

(2)将读出的数据加工成符合目标模式的数据.

(3)将加工后的数据写入目标数据库.

实现数据转换的基本方法有两种:

(1)解释型转换器法，见图 1.

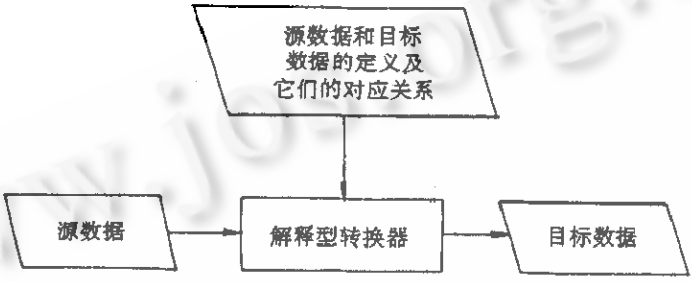


图 1 解释型转换器

(2)转换程序生成器法，见图 2.

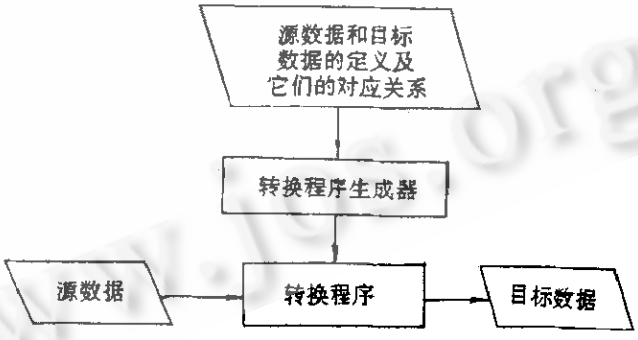


图 2 转换程序生成器

两种方法都以源数据和目标数据的定义及其对应关系的说明作为输入，进行数据转换，但实现的方法不同，其差别类似于程序设计语言的解释器和编译器的差别，前者易于实现，后者效率较高，具体选用哪种方法可根据实际情况确定.

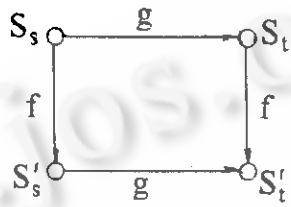
国外在几年前已做了一些数据转换的原型系统. CODASYL 存储数据的描述与转换工作组(SDDTTG)提出一种解释型数据转换系统的模型，它支持三种描述语言，分别在逻辑级和物理级对数据进行描述，并描述数据之间的对应关系，由解释型转换器根据这些描述进行转换[8].

IBM 的 San Jose 实验室的原型系统 XPRS 支持两种描述语言，DEFINE 和 CONVERT，其中，DEFINE 语言对数据进行描述，CONVERT 语言用来说明源数据应如何转换为目标数据．数据转换系统是一个转换程序生成器，它根据 DEFINE 语言描述的源和目标数据生成读出、写入程序，根据 CONVERT 语言描述的源和目标数据的对应关系生成数据加工程序．系统协调各程序的运行，完成数据转换[6][9][10]．

3. 应用程序的转换

应用程序的转换，首先要保持目标程序与源程序功能的一致性，这有以下两个含义：

(1)目标程序保持源程序对数据库的数据操作性质，即：



其中，f 为数据转换；g 为源应用程序；g′为目标应用程序；S_s 是源数据库的一个瞬象；S_t 是对 S_s 执行 g 后得到的源数据库的一个瞬象；s'_s 是 S_s 经过数据转换 f 得到的一个目标数据瞬象，S'_t 是 S'_s 执行 g′得到的目标数据库瞬象，它应该等于 S_t 经过数据转换 f 后得到的目标数据瞬象．

(2)目标程序保持源程序的数据处理性质．

在应用程序中，除了对数据库的操作外，还有一些语句对数据做求和、打印等处理．保持目标程序与源程序的数据处理性质一致，指的是目标程序中对某数据的处理可与源程序中对相应数据的处理在功能上相同，这样才能保证目标程序的用户所看到的询问及回答信息与源程序的用户所看到的一样．

实现应用程序的转换，首先要确定源程序的功能及语义，然后根据这些功能和语义，以及源数据模式，目标数据模式及它们之间对应关系的说明，分析出源程序存取的数据是什么，在目标 DBMS 上如何完成，生成一个目标 DBMS 上的应用程序来实现它，如图 3 所示．

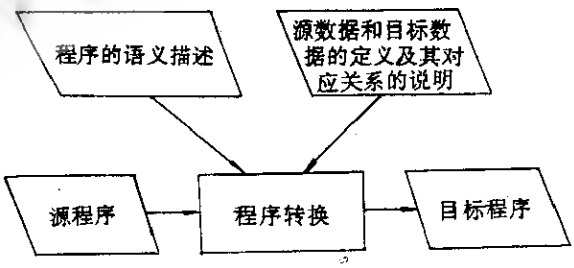


图 3 应用程序的转换

应用程序的特点导致了应用程序转换的复杂性：数据操作的对象不确定，变量做数据

操作命令的参数，使得数据操作的功能难于理解；程序中隐含使用数据的特点，数据操作在宿主程序中的嵌入结构复杂，使转换系统难于理解程序的功能；数据操作语言的级别不同，也给应用程序的转换带来了困难．所以，建立一个通用的，与 DBMS 无关的应用程序转换系统几乎是不可能的，一个可取的办法是建立一个半自动的转换系统，在人工的协助下完成程序的转换工作．

目前国外已有一些程序转换的方法．例如：DML 语句替换法，是在应用程序的执行时，截取各个 DML 语句，分析这个语句的功能，代之以适合于数据库新的逻辑结构的 DML 语句，完成原来 DML 语句的功能．这种方法只适用于同一 DBMS 上模式发生变化时对应用程序的转换．另一种方法是保留源程序不变，通过一个"桥程序"(实际上是一个双向的数据转换程序)，将目标数据库中源程序用到的那部分数据转换成源程序的形式，支持源程序的存取要求，使源程序以原来的方式运行，将得到的结果数据经过逆向映射，改变目标数据库．这两种方法共同的缺点是效率低，并且限制了对新的 DBMS 或新的数据模式的能力的使用．

现在，已有一些工作在研究建立通用的工具来自动地或半自动地修改或重写源应用程序，使用新的 DBMS 的优点来生成目标应用程序，以克服以上两种方法的缺点．

一种将 CODASYL 的 DML 语句转换成关系查询的方法是：根据查询操作的特点画出当前值流图，从中导出存取路径表达式，根据这个表达式生成关系查询，嵌入源应用程序中，代替源 DML 语句及有关的控制语句[3]．参考文献[8],[11],[12]给出了几种其它的转换方法．

§3. 系统的设计

我们设计了一个数据库转换系统，将 IMS 上的信息系统转换为 ORACLE 上的信息系统．

1. IMS 到 ORACLE 的模式转换

IMS 是层次型的数据库管理系统，其数据模式是由若干相关联的片段组成的一个层次结构，约束是：除根片段外的每个片段有且只有一个物理父母片段．

IMS 数据库分为物理数据库(简称 PDB)，和逻辑数据库(简称 LDB)．一个 PDB 是一个物理数据库记录型的全体值的有序集，其中的片段是实际存储的．LDB 是联合一个或多个物理数据库的部分片段型构成的层次结构，其中的片段来源于各物理数据库．根据 PDB 和 LDB 的特点，我们分别对它们进行转换：

对 PDB，生成相应的关系模式，方法是：

(1)为每个片段 S_i 建立一个关系 R_i，S_i 中的所有字段在 R_i 中都有相应的属性，R_i 的码等于 S_i 的码(根据大多数情况，不妨假设每个片段都有一个字段是唯一标识片段的码)．

(2)如果 S_j 是 S_i 的父母片段(目标物理和逻辑的父母)，则 R_j 的码作为 R_i 的一个属性．

使用这种方法，片段的所有字段在关系中有相应的属性，使转换后的关系模式保持了源 IMS 模式的所有数据；在子女片段所对应的关系中加上父母片段的码作属性，保持了源模式的层次结构．由于 ORACLE 中没有提供外码约束的功能，我们的模式转换不能将 IMS 模式上的约束直接转换到 ORACLE 的数据模式上，而是在程序转换中通过在目标

程序中加入对外码的检查来实现外码约束。

对 LDB，我们只记录它的层次结构和各片段与物理片段的对应关系，把这信息放在中间数据文件中，供应用程序的转换使用。

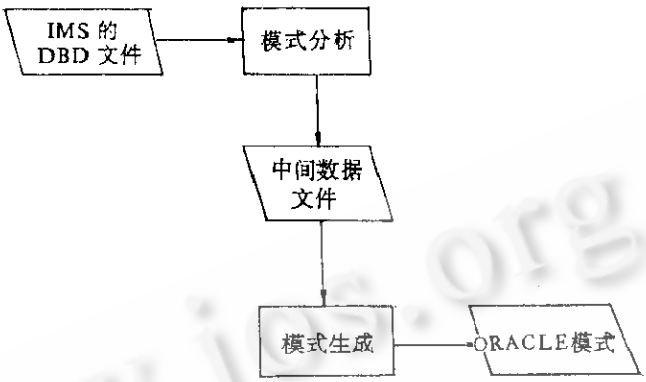


图 4 IMS 到 ORACLE 的模式转换

模式转换以 IMS 数据库的 PBD 文件为输入，经过模式分析，生成描述层次数据库的中间数据文件，并根据中间数据文件生成 ORACLE 的关系模式。因为我们的 ORACLE 模式是根据一定的规则，根据中间数据文件生成的，生成规则简单，实现方便，所以，不必记录源模式和目标模式的对应关系，也不必记录目标模式的描述。

下面我们通过一个例子说明模式转换：

图 5 是两个 IMS 的物理数据库，图 6 是转换后的关系模式。从图中可以看出，转换后的关系数据库中，为每个片段建立了一个关系，并在子女片段对应的关系中加入父母片段对应的关系的码作属性。保留了原来的所有信息。例如，在助教这个关系中，职工号和助教名两个属性记录了原有片段的两个字段，系号这个属性指出了原片段的父母片段对应的关系元组。

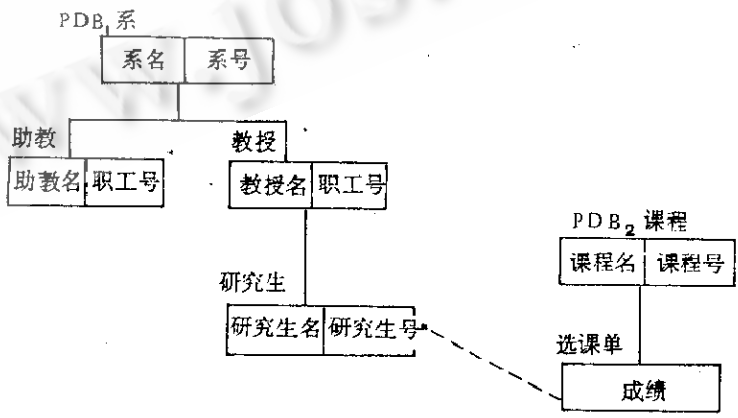


图 5 IMS 数据模式

系：系名，系号

助教：系号，助教名，职工号

教授：系号，教授名，职工号，

研究生：导师职工号，研究生名，研究生号

课程：课程名，课程号

选课单：课程号，研究生号，成绩

图 6 ORACLE 数据模式

2. IMS 到 ORACLE 的数据转换

我们的数据转换方法是专用的转换程序生成器法。"专用"指的是只实现 IMS 到 ORACLE 的数据转换。转换程序生成器根据中间数据文件中对 IMS 模式的说明以及 IMS 模式到 ORACLE 模式的对应规则，为 IMS 的每个物理数据库生成一个 COBOL 语言书写的转换程序和 ORACLE 的数据装入程序 ODL 使用的 ODL 控制文件。

这里的转换程序是一个 IMS 的应用程序，它使用 IMS 的用户界面，通过程序中嵌入的 DL／1 调用检索数据。作为 IMS 的应用程序，都要有一个 PSB 来说明程序可感知的片段，因此，转换程序生成器在生成转换程序的同时，也生成这个转换程序的 PSB。这个 PSB 感知物理数据库中的所有片段的所有字段，如果一个片段 S_i 有逻辑父母 S_j，则在 PSB 中将 S_i 和 S_j 的码说明成一个联合片段。

运行时，转换程序以深度优先的方式遍历层次模型，按层次序列码的顺序查询出每个片段值。使用这种查询方法，对查询到的每个值，其物理父母和逻辑父母的码值都已查到，将父母片段的码值和子女片段的值加工成子女片段对应关系的元组形式，放入顺序文件。对每个片段有一个顺序文件。在一个物理数据库的转换程序执行完后，顺序文件的数目与物理数据库中片段的数目一样多。

在一个 IMS 物理数据库中的数据全部转换成顺序文件后，执行 ORACLE 的实用程序 ODL，它根据 ODL 控制文件中说明的顺序文件中的数据与元组的对应关系，将顺序文件装入 ORACLE 数据库。

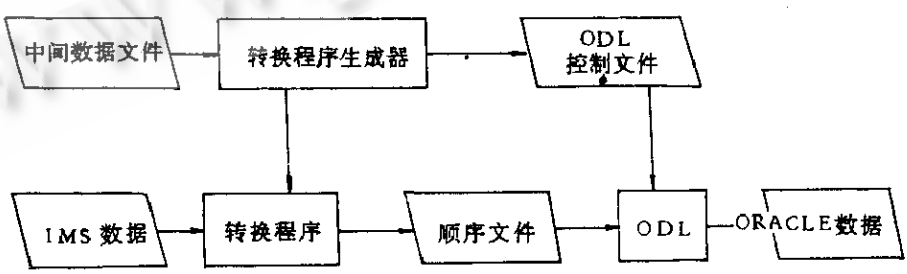


图 7 IMS 到 ORACLE 的数据转换

3. IMS 应用程序的转换

我们的转换系统将 IMS 上 COBOL 语言书写的应用程序转换为 ORACLE 上

COBOL 语言书写的应用程序(称为 Pro COBOL 程序).

IMS 的应用程序使用嵌入的 DL/1 语句，调用数据库接口，根据数据库的当前值一次对一个片段进行处理.

ORACLE 上的应用程序使用基于游标的，一次一个元组的 SQL 界面进行数据库操作，这种界面的使用方式是:

查询: 为查询语句定义一个游标. 打开游标，执行与游标相应的查询，将查询结果准备好，游标指向查出的第一个元组; 对游标使用 FETCH 语句，取出一个元组，放入宿主变量，同时将游标指向下一个元组; 查询结束时关闭游标. 这样，虽然数据的存取仍然是非过程型的，但对数据处理的方式已变成过程型的了.

更新: 先将元组放入宿主变量，以它为参数调用更新操作，完成对数据库的更新.

由于 ORACLE 数据库的级别比 IMS 数据库的高，层次数据库上的数据操作的一些功能在 ORACLE 的 SQL 语言中不能实现. 为此，我们给出"可转换"的 IMS 应用程序应满足的条件:

* 程序是结构化设计的;
* DL/1 调用必须有 SSA，且不能使用 F，L 等命令码;
* 不允许使用数据库的出错状态值;

对应用程序的这些限制并不过分，太多数程序都能满足，因此，这些限制并不影响系统的实用性. 首先,"程序是结构化设计的"这一要求是合理的; 其次，根据实际应用要求所写的数据操作语句一般都能满足第二条限制; 在信息系统中运行的程序，都已通过了调试，一般不使用数据库的出错状态值，只使用数据库的正常状态值和某些例外状态值，所以，大多数程序能满足第 3 条限制.

为使被转换的应用程序满足上述条件，先检查应用程序，找出那些不满足上述条件的语句，由人工改写，使之成为可转换的程序.

应用程序的转换分三步进行:

(1)将源程序对数据库数据的使用说明转换成 Pro*COBOL 程序中对相应的关系数据的说明:

IMS 上的应用程序，都有一个相应的程序说明块 PSB，说明程序中可使用的片段及使用方式. ORACLE 的 Pro*COBOL 程序是调用 SQL 语句，将查出的元组放到"宿主变量"中或根据"宿主变量"中的数据修改数据库. 这里的宿主变量不同于一般的程序变量，它既可以在宿主程序语句中使用，也可以在嵌入的 SQL 语句中使用，需要在数据库的 DECLARE 节中说明. 目标 Pro*COBOL 程序中用到的关系只可能是源 COBOL 程序用到的片段对应的关系，所以，我们根据 COBOL 程序的 PSB，生成目标 Pro*COBOL 程序中的宿主变量说明.

(2)确定过程部的控制结构

DL/1 语句是导航式的，一次对一个片段值操作，要通过循环体中的 DL/1 调用完成对满足某个条件的所有值的操作. 基于游标的 SQL 界面的使用方式是，先用打开游标语句将满足条件的所有元组值准备好，然后在循环体中用 FETCH 游标语句一次取出一个元组处理. 因此，需要确定 COBOL 源程序中 DL/1 语句是嵌入在哪个循环体中的，以决定目标程序中打开，关闭游标的位置.

我们限定了程序是结构化设计的，所以循环只能用PERFORM…UNTIL语句控制完成．我们只需按程序的控制结构扫描一遍，就能确定DL／1语句所在的循环体．例如：扫描到一个语句：PERFORM A THRU B UNTIL COND-1

下面扫描标号A到B这段程序遇到的DL／1语句就是这个PERFORM语句控制循环的。

(3)将DL／1操作转换为SQL操作。

转换数据操作，首先要知道它的功能，这包括以下两方面：

①数据操作的类型．DL／1语句参数中的功能码说明了操作的类型。

②数据操作的对象及操作对象应满足的条件．根据"可转换"的应用程序的限制，每个DL／1操作必须带有SSA．在SSA中说明了这个操作的对象及它们应满足的条件。

将DL／1语句转换成SQL语句，重要的一步是将DL／1的SSA参数转换为SQL的WHERE子句．转换时，不仅要将片段上的限制条件转换为元组上的限制条件，还要加上联接条件，体现关系间的层次联系．例子见图8。

GNP型的操作根据父母片段的当前值查询，因此，在转换后的WHERE子句中还要加上对父母片段对应的元组的限制条件．例如：

GNP查询语句的SSA是：研究生

转换后的WHERE子句是：WHERE研究生．导师号＝已查到的教授.职工号

DL／1的SSA　　WHERE子句

系(系号＝'08')　　系.系号＝'08'

职工(职工号＝'800')　　职工.职工号＝'800'　} 元组上的限制条件

职工.系号＝系．系号 } 联接条件

图8　SSA到WHERE子句的转换

数据操作的转换分以下三类：

(1)一个DL／1操作转换为一个SQL操作，DL／1语句与SQL语句的对应关系是：

DL／1语句	SQL语句
GU型查询语句	打开相应的查询语句的游标 FETCH游标操作 关闭游标
GN, GNP型查询语句	FETCH语句 控制所在循环的PERFORM语言前 (后)加游标的打开(关闭)语句
ISRT语句	INSERT语句
REPL语句	UPDATE语句

(2)一个 DL/1 语句对转换成一个 SQL 语句

IMS 上的一种常见的查询方式是用一个 GU、GN 语句对(其中 GN 在循环体中执行)，查询满足某一条件的所有片段值。我们将这个语句对转换成一个 SQL 语句，查询满足这个条件的元组。例如：原应用程序段为：

GU 型查询;
PERFORM A THRU B UNTIL 条件 1:
A. 对数据的处理;
GN 型查询;
B.

转换后的程序段为

定义游标 C1;
OPEN C1;
FETCH C1;
PERFORM A THRU B UNTIL 条件 1;
CLOSE C1:
A. 对数据的处理;
FETCH C1;
B.

(3)一个 DL/1 语句转换为多个 SQL 语句

在 IMS 数据库中删除一个片段，系统会自动将它的子女片段删除。在 ORACLE 数据库中没有完成这项工作的机制，要由应用程序完成对子女片段对应元组的删除，这样，一个 DL/1 的删除语句就转换成了一个 SQL 删除语句序列。

Pro*COBOL 程序与 ORACLE 的数据是通过宿主变量交换的，源 COBOL 程序与 IMS 的数据交换是通过 I/O 区进行的。我们保留源程序中的所有数据处理语句，并在 SQL 的查询语句后和更新语句前加上 MOVE 语句实现宿主变量和 I/O 区的数据交换，这样就保证了目标程序与源程序的数据处理性质一致。

对 IMS 应用程序所用的数据库状态值，我们也做了相应的转换。例如，在查询不到满足 SSA 的片段时，IMS 返回的数据库状态值是'GE'，我们将这个状态值转换为 ORACLE 上的 Whenever not found 状态。

§4. 小结

IMS 是 IBM 公司早期推出的层次型 DBMS，它的使用范围很广，有大量用户在上面建立了大型的信息系统，国内也有不少这样的信息系统。现在，IBM 的机器上又开发了新的关系型 DBMS，它的数据独立性强，用户界面好等特点使它优于层次型 DBMS。因此，用户希望能将原有的 IMS 上的信息系统"搬"到关系数据库上使用。根据这一应用背景，我们研制了这个数据库转换系统，将 IMS 上的信息系统转换为 ORACLE 上的有同样功能的信息系统。

在我们设计的数据库转换的算法中使用了较为复杂的数据结构，因此，我们选用数据

结构丰富、功能较强的 C 语言来实现。由于采用的方法是生成 COBOL 语言的程序，所以系统有一定的适应性。今后我们要进一步将系统实用化。

参考文献

[1] D. C. Tsichristzis & F. H. Lochovsky,《DATA MODEL》.

[2] 萨师煊、王珊.《数据库系统概论》.

[3] R. H. Katz & E. Wong,《Decompiling CODASYL DML into Relational Queries》, ACM TODS, Vol.7, No.1, pp. 1–23.

[4] V. Y. Lum et al,《A General Methodology for Data Conversion and Restructuring》, IBM J. RES. DEVELOP, 1976. 9, pp.483–497.

[5] J. P. Fry et al.,《An Accessment of the Technology for Data and Program RelatedConversion》.

[6] N. C. Shu, et al,《EXPRESS: A Data Extraction, Processing and Restructing System》, ACM TODS, Vol.2, pp 134–174.

[7]《Stored–Data Description and Data Translation: A Model and Language》, Inform. Systems, Vol. 2, pp. 95–148.

[8] Su, Y. W. Stanley,《Application Program Conversion Due to Database Changes》,Proc. of the 2nd International Conference on VLDB.

[9] Housel, et al.,《DEFINE: A Non–Procedural Data Description Language for Defining Information Easily》, Proc. of 1975 ACM Pacific Conference, pp. 62–70.

[10] Shu, et al,《CONVERT: A High–level Translation Definition Language for Data Conversion》, Comm. ACM Vol.18, No.10, pp. 557–567.

[11] Su, S. Y. W and B. J. Liu,《A Methodology of Application Program Analysis and Conversion Based on Database Semantics》, Proc. International Conference on Management of Data, 1977, pp. 75–87.

[12] IMS 用户手册.

[13] ORACLE 用户手册.