MTA SZTAKI Közlemények, 35/1986 p.105-125

ЛОРКА - РАСШИРЕНИЕ ПАСКАЛЯ К РЕЛЯЦИОННОЙ МОДЕЛИ БАЗ ДАННЫХ

М. Катриб Мора, Е. Квесада Орозцо, Й. Баш Байард, Е. Оберт Вазквез

Кафедра вычислительной техники, Гаванский Университет, Куба

ВВЕДЕНИЕ

Файлы являются единственным средством, предоставляемым языком ПАСКАЛЬ для хранения внешней по отношению к программам и, следовательно, годной для использования после окончания их выполнения, информации. Этого недостаточно для работы систем управления базами данных /СУБД/, т.к. любой программе, использующей файл, созданный другой программой, необходимо знать точное определение типа данных, содержащихся в нем, о чем ПАСКАЛЬ не записывает никаких сведений. Любое изменение в структуре файла требует перепрограммирования и перекомпиляцию всех программ, его использующих.

С другой стороны, организация и управление внешней памятью осуществляется операционной системой /ОС/, под управлением которой работает компилятор с языка высокого уровня /будь это ПАСКАЛЬ или любой другой/, что вынуждает СУБД, ориентированные на использование файлов, ограничиваться возможностями ОС. По этой причине очень часто в таких системах информация, содержащаяся в базах данных, не защищена и доступна "чужим" программам.

Компонентны таких файлов должны иметь, как правило, одинаковые размеры. Этот факт ограничивает более динамическое использование памяти, поскольку не все виды информации, встречающиеся в приложениях, имеют такие характеристики.

ЧТО ТАКОЕ ЛОРКА

ЛОРКА - это СУБД, основанная на модульном расширении ПАСКАЛЯ. Это расширение состоит в добавлении новых типов, переменных, процедур и функций, составляющих удобный интерфейс для работы

с понятиями реляционной модели и в то же время позволяющих скрыть от пользователя особенности работы системы.

ЛОРКА предоставляет пользователю:

- Широкий набор типов данных, облегчающих создание баз данных /БД/, особенно в области научных исследований.
- Гибкие интерактивные средства для ввода и модификации данных посредством экранных редакторов.
- Быструю селективную выборку информации, хранящейся в БД, на основе паскалевидных формул, обобщающих нотацию реляционной алгебры.
- Возможность хранения этих формул внутри самой БД, чтобы они могли быть применены без необходимости повторного ввода.
- Возможность организации отношений, что гарантирует доступ к любой отдельной записи.
- Большую экономию памяти на основе использования различных динамических форм представления информации.
- Возможность быстрых и надежных копий БД и передачи информации между ними.
- Возможность взаимодействия с файловой системой ОС и передачи информации от Паскаль-файлов в БД и обратно.

ТИПЫ ДАННЫХ

Множество возможных значений любого атрибута, а также допустимые над ними операции характеризуют то, что будем называть типом данных /ТД/ атрибута.

Типы данных в ЛОРКе разделяются на: простые, текстовые, кодовые и структурные. Простыми типами являются: целый, байтовый, вещественный, а также цепочки, даты и литералы. Структурными ТД являются векторы и множества элементов любого простого типа.

ЦЕЛЫЙ Т.Д.

Целый Т.Д. соответсвует типу integer языка ПАСКАЛЬ.

БАЙТОВЫЙ Т.Д.

Это тип отвечает определению:

```
type byte=0..255
```

ВЕЩЕСТВЕННЫЙ Т.Д.

Использование вещественных чисел в формате с плавающей запятой характерно для всех научных применений. Включение в ЛОРКу числовых данных в виде целых и вещественных величин позволяет использовать систему в области научных исследований, а также достичь более компактную форму представления данных в целях экономии памяти. Этот тип подобен стандартному типу real языка ПАСКАЛЬ.

ДАТЫ

ЛОРКА включает предопределенный тип дата. Константа этого типа записывается в виде:

<day> : <month> : <year>

где <day> обозначает целую константу в ранге 1.31, <month> целую константу в ранге 1.12, и <year> - положительное целое.

П р и м е р ы :

12 : 3 : 78 30:11:84 1:1:1985 12:3:1978

Соответствующий тип языка ПАСКАЛЬ определяется как:

```
date = packed record
         day   : 0 .. 255;
         month : 0 .. 255;
         year  : 0 .. maxint
       end
```

ЛИТЕРАЛЫ

Во многих приложениях необходимо манипулировать атрибутами, значения которых отображают качественными характеристиками, неподдающимися количественному измерению.

В большинстве существующих СУБД, этот тип данных представляет-

ся в виде цепочек символов /что является неэкономичным, поскольку должна выделяться память для целой цепочки при каждом появлении одного и того же значения/, либо представляется в виде чисел, что заставляет пользователя заботиться о кодировании и декодировании этих переменных.

ЛОРКА представляет пользователю тип ЛИТЕРАЛ. Константа этого ТД может быть цепочка символов /максимальная длина которой определяется при создании БД/. Значения атрибутов, отвечающих этому ТД, задаются пользователем в виде цепочек, а предсталяются они внутри системы кодированным способом, что приводит к большей экономии памяти и эффективности во времени.

ЛОРКА также включает предопределенное значение НЕИЗВЕСТНОЕ, которое воспринимается по умолчанию как значение любого атрибута этого ТД, который не был инициализирован явным образом.

> Перечисленные типы Паскаля являются недостаточными для того, чтобы отражать суть этого понятия, т.к. Паскаль не сохраняет в памяти перечисленные идентификаторы.

В Паскаль-расширении этот тип должен считаться частным и может быть использован только через собственные ресурсы расширения:

```
type literal = private
```

ЛОРКА располагает, кроме того, двумя процедурами для преобразования цепочек языка Паскаль в литералы и наоборот.

```
function Str_to_lit(lit_name:string;
                    var lit:literal):boolean;
procedure Lit_to_str(lit:literal;
                    var lit_name:string)
```

В отличии от перечисляемых типов Паскаля, включение литералов в БД может осуществляться динамическим образом с помощью процедуры:

```
procedure Insert_lit(lit_name:string)
```

Литералы можно переименовать с помощью процедуры:

```
procedure Rename_lit(old_name,new_name:string)
```

Процедура

```
procedure Literals_to_file(file_name:string)
```

создает текстовый файл, содержащий имена всех литералов любой БД.

ЦЕПОЧКИ

Для тех атрибутов, диапазон значений которых должен отображаться последовательностью символов и которые не могут быть представлены литералами, т.к. невозможно перечислить совокупность всех возможных значений, в ЛОРКу включен тип данных string.

Этот ТД подобен цепочечному типу языка Паскаль. Однако в ЛОРКе значения этого типа хранятся внутри БД согласно своей текущей длине, а не какой-то максимальной.

ВЕКТОРНЫЕ СТРУКТУРЫ

Во многих научных приложениях, например, для математико-статистической обработки, необходимо манипулировать данными, предоставленными в виде векторов вещественных чисел. Однако, в реляционной модели БД эта форма представления неэффективна, поскольку заставляет пользователя задавать имя каждому из атрибутов.

В ЛОРКу включена возможность определить векторы из элементов, принадлежащих любому из простых ТД /кроме цепочечного/. Длина вектора определяется пользователем при создании БД.

Таким образом, оперируя одним атрибутом /и одним именем/ можно иметь доступ к целому вектору. Когда требуется обращение к конкретному элементу вектора, этого можно добиться с помощью индекса, как это делается в большинстве языков программирования.

Значение типа ВЕКТОР представляется в системе в виде:

$$[\langle value_1 \rangle, \langle value_2 \rangle, \ldots \langle value_n \rangle]$$

где каждое value обозначает значение простого типа. Соответствие между структурными типами этого класса ЛОРКи и Паскаля следующее:

```
     array[1..n] of integer   - для целочисленных векторов
     array[1..n] of real      - для векторов вещественных чисел
  packed array[1..n] of byte  - для байтовых векторов
  array [1..n] of literal     - для векторов, составленных из
                                литералов.
```

СТРУКТУРА ТИПА МНОЖЕСТВА

В ЛОРКу включены в качестве типов атрибутов множества, состоящих из значений любого из простых типов. Атрибут, определенный таким образом, может иметь своим значением множество значений базового типа.

Любое значение этого типа обозначается в форме:

$$\{<value_1>, <value_2>, \ldots, <value_n>\}$$

где каждое value обозначает значение простого типа.

Множество в ЛОРКе не имеют эвивалента на ПАСКАЛе. Работа со значениями этого типа может осуществляться только посредством функций и процедур, предоставленных системой.

ТЕКСТОВЫЙ ТИП

Этот ТД используется для представления в БД текстов большой длины, такие как документы или резюме документов. Значения атрибутов текстового типа могут быть показаны на дисплее для модификации с помощью экранного редактора.

Текстовый тип может быть использован также для представления формул реляционного исчисления. Таким образом, пользователь может хранить свои формулы внутри собственной БД, чтобы они могли быть использованы и/или модифицированы в любой момент.

Этот ТД не имеет аналога на языке ПАСКАЛЬ и дожен считаться частным /private/. По этой причине его можно использовать толь-

ко через системные функции.

```
type db_text = private.
```

КОДОВЫЙ ТИП

Результат компиляции текста, соответствующего формуле реляционной алгебры, является значением кодового типа. Такие значения не могут быть распечатаны, т.к. они соответствуют внутреннему представлению формулы отбора.

Хотя ЛОРКА представляет общую интерактивную версию, текстовые и кодовые значения используются для разработки новых версий.

ОПЕРАЦИИ С БД

Система располагает следующими ресурсами для работы с БД:

а/ Создание БД

Эту операцию можно выполнить с помощью

```
procedure Define_DB(db_name:string;
                    protection_key:string;
                    size:0..maxint);
```

При этом определяется пустая БД: без отношений и с нулевым словарем.

б/ Открытие БД

```
procedure Use_DB(db_name:string;
                 protection_key:string)
```

Открывает для последующего использования базу данных. Все операции, которые будут рассматриваться дальше, относятся к открытой БД и поэтому не делают явную ссылку к ней.

Для того, чтобы узнать количество свободной памяти в БД используется:

```
function Mem_in_DB:integer;
```

которая возвращает колическо свободных страниц /1 страница = 1024 байтов/.

ОТНОШЕНИЯ В ЛОРКА-ПАСКАЛе

Отношения представляются в ЛОРКе с помощью типа.

Этот тип не имеет эквивалента на ПАСКАЛе и поэтому с объектами такой натуры можно работать только через возможности ЛОРКи.

```
type relation = private
```

Отношения могут находиться в одном из двух состояний или режимов работы: query - если над ней не будут выполняться никаких преобразований /информация будет только считываться/, или actualization, если будут производиться модификации.

```
rel_operation = (query, actualization)
```

ОПРЕДЕЛЕНИЕ ОТНОШЕНИЯ

```
procedure Define_rel(rel_name:string)
```

Для определения реляционной схемы отношения необходимо определить его атрибуты. Это делается с помощью процедур Define_atrib,Append_atrib Copy_structure. Для того, чтобы охарактеризовать типы значений атрибутов задаются:

```
type Data_type = (Int_type,Byte_type,Real_type,Literal_type,
                  Date_type,String_type,Text_type,Code_type,
                  Int_vector,Byte_vector,Real_vector,
                  Date_vector,Literal_vector,
                  Int_set,Byte_set,Real_set,Literal_set,
                  Date_set,String_set);
type atribute = private
```

Тип надо рассматривать как собственный тип ЛОРКи и может быть использован только через следующие операции:

а/
```
procedure Define_atrib(rel_name:string;
                       atrib_name:string;
                       atrib_type:data_type;
                       length_vector:0..maxint)
```
включает атрибут в схему отношения.

б/ Если пользователь желает добавить в схему атрибуты, уже

существующие /определенные в других отношениях/, используются:

```
procedure Append_atrib(rel_name:string;
                       atrib_name:string)
```

в/
```
function Str_to_atrib(atrib_name:string;
                      rel_name:string;
                      var atrib:atribute):boolean
```

Эта функция определяет, является ли atrib_name именем атрибута данного отношения и возвращает этот атрибут в переменной atrib.

г/ Тип значений атрибута может определяться функцией:

```
function atrib_type(atrib:atribute):data_type
```

д/ Имя атрибута можно определить процедурой:

```
procedure Atrib_to_str(atrib:atribute;
                       var atrib_name:string)
```

е/ Операция

```
procedure Rename_atrib(old_name:string;
                       new_name:string)
```

позволяет модифицировать имя атрибута.

ж/
```
procedure Copy_structure(source_rel_name,
                         destination_ret_name:string)
```

Добавляет к реляционной схеме второго отношения те атрибуты первого /исходного/ отношения, не являющиеся атрибутами второго отношения.

ОБЩИЕ ОПЕРАЦИИ НАД ОТНОШЕНИЯМИ

ЛОРКА включает операции, действующие одновременно над целым отношением:

а/
```
function Delete_rel(rel_name:string):boolean
```

Эта функция стирает /убирает/ отношение из БД и освобожда-

ет им занятую память.

б/ function Sort_rel(rel_name:string;
atrib:atribute):boolean

Производит сортировку отношения по заданному атрибуту.

в/ procedure Tuples_rel(rel_name:string;
var total_tuples:integer)

Возвращает количество кортежей /записей/ названного отношения.

г/ function Total_atribs(rel_name:string):integer

Возвращает общее количество атрибутов отношения/в случае ошибки возвращает нуль/.

д/ procedure Copy_rel(source_rel_name,
destination_rel_name:string)

Создает копию исходного отношения с именем второго отношения. Если раньше существовало отношение с именем destination rel_name, то прежнее его содержимое уничтожается.

е/ procedure Union_rels(rel1_name,rel2_name:string)

Создает новое отношение, являющееся результатом соединения отношений rel1_name, rel2_name.

ж/ procedure Union_rels(rel1,name,rel2_name:string)

Оба отношения должны иметь одинаковую реляционную схему. Эта процедура создает отношение, являющееся объединением rel1_name, rel2_name.

з/ procedure Concat_rels(rel1_name,rel2_name:string)

Оба отношения должны иметь одинаковую реляционную схему. Создается отношение, являющееся результатом добавления /т.е. могут повторяться кортежи/ к отношених rel1_name кортежи отношения rel2_name.

и/ Процедура
procedure Rename_rel(rel_name,new_name:string)
позволяет переименовать отношение.

ОПЕРАЦИИ ДЛЯ ОБРАБОТКИ И МОДИФИКАЦИИ ОТНОШЕНИЙ

а/ Открытие отношения

Операции, о которых будет говориться дальше, позволяют обработать данные, содержащиеся в любой БД, с программ, работающих в системе.

```
procedure Open_rel(rel_name:string; mode:rel_operation;
                   var rel:relation)
```

связывает имя отношения со значением типа relation, который в дальнейшем будет использоваться при работе с отношением.

Если отношение открывается в режиме query, информация, имеющаяся в отношении, может быть только считана; если отношение открывается в режиме modification, то информация готова как для считывания, так и для записи /модификации/.

б/ Текущий кортеж

В любой момент над каждым открытым отношением возможен доступ только к одному кортежу, называющемуся текущим. Таким образом, обработка отношения производится путем просматривания его с помощью текущего кортежа.

После открытия отношения текущим является самый первый кортеж.

Процедура

```
procedure Get_tuple(rel:relation)
```

делает текущим следующий кортеж названного отношения.

Если его не существует, то булевая функция Eorel(rel) возвращает true, и назначение текущего кортежа становится неопределенным.

Функция

```
function Eorel(rel:relation):boolean
```

возвращает значение true, если задана Get_tuple при текущем кортеже, равным последнему кортежу отношения, или если открывается пустое отношение. В других случаях возвращает

false.

Процедура

procedure Reset_rel(rel:relation)

позиционирует уже открытое отношение в первом его кортеже.

в/ Прямой доуступ к кортежу

```
function Find_tuple(var Pascal_data;
                    rel:relation;
                    atrib:atribute):boolean
```

Ведет поиск, начиная с текущего кортежа, следующего кортежа отношения, имеющего значение атрибута atrib, равным параметру Pascal_data. Если кортеж существует, то он становится текущим и функция возвращает true; в противном случае изменений не происходит и функция возвращает false.

г/ Стирание кортежей

procedure Delete_tuple(rel:relation);

Применима только в случае, если отношение rel открыто в режиме actualization. Физически убирает текущий кортеж.

д/ Копирование кортежа

procedure Copy_tuple(sourece__el,destination_rel:relation)

Копирует текущий кортеж исходного отношения после текущего кортежа отношения назначения и делает его текущим. При этом реляционные схемы обоих отношений должны совпадать.

е/ Добавление кортежа

Вставляет нулевой кортеж /каждый его атрибут инициализируется нулевым значением соответствующего ТД/ после текущего кортежа отношения. Этот нулевой кортеж может быть потом отредактирован с помощью предоставленных системой возможностей. Если eorel имеет значение true, то нулевой кортеж становится текущим и eorel переводится в false.

ж/ Редактирование кортежей

```
procedure Edit_tuple(rel:relation; mode:edit_mode);
where edit_mode=(new_tuple, old_tuple)
```

При обращении к этой процедуре в режиме old_tuple показывается на дисплее текущий кортеж отношения и вызывается редактор кортежей ЛОРКи. Результат редактирования заменяет текущий кортеж /отношение должно быть открытым для модификации/. При обращении к ней с параметром new_tuple возможно диалоговое редактирование нового кортежа.

з/ Соединение кортежей

```
function Join_tuples(rel1,rel2:relation)
```

Возвращает true, если текущие кортежи отношений rel1, rel2 удовлетворяют усвловиям соединения.

```
procedure Append_tuples(rel1,rel2,rel3:relation)
```

Вставляет новый кортеж в rel3, что соответствует добавлению текущему кортежу rel1 значений атрибутов отношения rel2, которые не являются атрибутами rel1.

ПЕРЕДАЧА ЗНАЧЕНИЙ МЕЖДУ ЛОРКОЙ И ПАСКАЛЕМ

Следующие операции работают над текущим кортежом отношения и передают ПАСКАЛю значение определенного атрибута кортежа или модифицируют /паскалевой константой/ значение одного атрибута.

а/ Передача значений в Паскаль

```
procedure Get_value(rel:relation;
                    atrib:atribute;
                    var Pascal_data)
```

Возвращает через параметр Pascal_data соответствующее значение языка Паскаль.

В случае, когда атрибут является вектором, фактически параметр должен иметь соответствующий тип array языка Паскаль. Для того, чтобы иметь доступ к отдельным компонентам без необходимости передачи всего вектора включена операция:

```
procedure Get_item_vector(rel:relation;
                          atrib:atribute;
                          index_item:integer;
                          var Pascal_data)
```

которая возвращает компоненту index_item вектора. Для того, чтобы работать со значениями множественных атрибутов даются следующие операции:

1.
```
function Cardinal_set(rel:relation;
                      atrib:atribute):integer;
```

   Возвращает количество элементов множества с именем atrib текущего кортежа отношения rel.

2.
```
procedure Get_item_set(rel:relation;
                       atrib:atribute;
                       index_item:integer;
                       var Pascal_data)
```

   Возвращает в параметре Pascal_data элемент index_item множественного атрибута atrib отношения rel.

3.
```
function Memeber(rel:relation;
                 atrib:atribute;
                 var Pascal_data :boolean)
```

   Эта функция является предопределенной для достижения большей эффективности работы системы, т.к. она является основой всех множественных операций.

б/ Копирование значений одного атрибута в другой

```
procedure Copy_value(source_rel:relation;
                     source_atrib:atribute;
                     dest_rel:relation;
                     dest_atrib:atribute)
```

С помощью этой процедуры возможно формирование кортежей, на основе кортежей других /или даже того же/ отношений без необходимости передачи посредством Паскаль-значений.

в/ Передача значений из ПАСКАЛЯ в ЛОРКУ

Следующая операция является обратной по отношению к Get_value:

```
procedure Put_value(rel:relation;
                    atrib:atribute;
                    var Pascal_data)
```

Если атрибут есть вектор, тогда фактически параметр Pascal_data должен быть соответствующего векторного типа.

Для модификации элементов множества имеется операция

```
procedure Put_item_set(rel:relation;
                       atrib:atribute;
                       var Pascal_data)
```

которая включает элемент Pascal_data во множество значений атрибута atrib.

```
procedure Put_item_vector(rel:relation;
                          atrib:atribute;
                          index_item:integer;
                          var Pascal_data)
```

Присваивает значение Pascal_data в качестве index_item компоненты вектора значений атрибута atrib в текущем кортеже.

```
procedure Rem_item_set(rel:relation;
                       atrib:atribute;
                       var Pascal_data)
```

Убирает из множества значений атрибута atrib элемент, равным параметру Pascal_data.

ТЕКСТЫ В ЛОРКА-ПАСКАЛЕ

а/ Редактирование текстов

Создание и модификация атрибутов этого типа может выполняться только через

```
procedure Edit_text(source_text:db_text;
                    var dest_text:db_text;
                    mode:edit_mode)
```

которая соответствует экранному редактору ЛОРКИ.

Если программист больше не будет использовать значение текстового типа, то его ответственностью является освобождение запятой текстом памяти. С этой целью включена операция:

```
procedure Free_text(text_to_deallocate:db_text)
```

б/ Передача текстов между БД и файлами Паскаля

```
procedure File_to_text(source_file:string;
                       var dest_text:db_text)
```

source_file должен являться именем текстового файла операционной системы, под управлением которого работает Паскаль. Эта процедура передает содержимое файла в ЛОРКу.

```
procedure Text_to_file(source_text:db_text;
                       dest_file:string)
```

Передает текст source_text из ЛОРКи в текстовый файл.

Этим файлом могут являться тоже АЦПУ или дисплей.

КОДЫ В ЛОРКЕ

Текст, полученный с помощью предыдущих операций, может отражать формулу реляционной алгебры. Эта формула может быть преобразована к виду, более близкому к машинному языку с помощью

```
procedure Compile(text_formula:db_text;
                  var code_formula:code;
                  var OK:boolean)
```

Этот код генерируется в оперативной памяти. Для записи его в БД используется

```
procedure Put_code(icod:code; var cod:db_code)
```

Значения типа db_code могут быть включены в кортежи БД через соответствующее присваивание с помощью put_value. Для освобождения внутренней памяти, занятой кодовым значением используется процедура

```
procedure Free_code(cod:code);
```

Для освобождения внешней памяти используется

```
procedure Free_dbcode(cod:db_code)
```

Кроме того, ЛОРКА представляет процедуру garbage_collector, которая собирает всю занятую память, которая больше не будет использоваться.

ВЫПОЛНЕНИЕ КОДОВ

Результат компиляции формулы, т.е. кодовое значение, может быть применено к текущему кортежу одного /или двух, если запрос подразумевает соединение/ отношения с помощью

```
function Apply var(code_formula:db_code;
                   rel1,rel2:relation):boolean
```

Если кортеж /кортежи/ удовлетворяет /ют/ условию, функция возвращает true.

Если пользователь желает выполнить ранее скомпилированный запрос, который хранится вБД в виде значения db_code, это значение необходимо загрузить в оперативную память с помощью

```
procedure Load(ext_cod:db_code;
               var internal_code:code)
```

Включение в ЛОРКА-ПАСКАЛЬ операций edit_text, compile, apply делает возможным создание и применение запросов во время выполнения прграммы без необходимости их явного включения в структуру программы, что приводит к высокой степени независимости и гибкости. Таким образом, можно хранить даже внутри самой БД запросы /в форме текстов и/или кодов/ и задать их в течение времени прогона любой программы.

Пример

Допустим в БД научно-технической информации имеется отношение USERS /Пользователи/ с атрибутами.

```
Name(string)
Dpt (literal)
Key_wards (db_code)
```

и имеется отношение PAPERS /Документы/ с атрибутами

```
Title   (string)
Author  (string)
Date    (date)
Journal  (literal)
Subject  (set of literals)
```

Атрибут key_words отражает профиль библиографических интересов пользователя, например:

```
With PAPERS do(Journal = CACM) and
{PASCAL,ADA} disjoint Subject
```

который выбирает те кортежи отношения PAPERS из журнала CACM, имеющие тематикой любое из двух значений PASCAL или ADA.

Если бы мы хотели распечатать названия работ, интересующих преподавателей департамента Вычислительной Техники /Computer Sciences/, то можно было бы поступить так:

```
var dept,ComputerSciences:literal;
    key_words_atrip,title_atrib,dept_atrib:atribute;
    atrib_type:data_type;
    user_kw:db_code; cod_user_kw:code;
    title:string;
    null_rel,users_rel,papers_rel:relation;

begin
  str_to_atrib(d'dpt','USERS',dept_atrib,atrib_type);
  str_to_atrib('key_words','USERS',key_words_atrib,atrib_type);
  str_to_atrib('title','PAPERS',title_atrib,atrib_type);
  if not st_to_lit('ComputerSciences',ComputerSciences) then
    writeln('Error ComputerSciences must be a literal');
  open_rel ( 'USERS',query,users_rel);
  open_rel ( 'PAPERS',query,papers_rel);
  while not eorel(users_rel) do
```

```
begin
  get_value(users_rel,dept_atrib,dept);
   { Get user department }
  if dept=ComputerSciences then
    begin
      get_value(user_kw,users_rel,key_words_atrib);
       { Get user key words interests }
      load(user_kw,cod_user_kw):
      while not eorel(papers_rel) do
       begin
         if apply cod user kw,papers_rel,null_rel) then
          begin
            get value(title,papers_rel,title_atrib);
              { Get a title from a paper }
            writen(title)
          end;
         get_tuple(papers_rel)  { Get new paper }
       end;
      free_code(cod_user kw);
      reset(papers_rel)
    end;
  get_tuple(user_rel) { Get new user }
 end
end
```

ЗАКЛЮЧЕНИЕ

Интерактивная версия ЛОРКИ разработана, используя вышеописанное расширение Паскаля, что облегчает ее использование, если требуется только классические операции реляционного исчисления. Включение различных предопределенных типов данных и возможности создания и обработки баз данных с языка ПАСКАЛЬ создает благоприятные условия для разработки конкретных рабочих сред.

Включение цепочек перемнной длины, литералов и множеств обеспечиваеют большую гибкость использования и экономию памяти по сравнению с другими системами, работающими на основе полей формированной длины.

Описанные возможности не представляются пользователям в других СУБД, имеющих узкоспециализированные языки, очень бедные по сравнению с современными языками программирования как Pascal.

Так как ПАСКАЛЬ не дает никаких механизмов для определения частных типов, настоящее расширение требует дисциплинированного применения со стороны программистов, которые должны использовать эти ТД только через предоставленные функции и процедуры. Препроцессор или новый компилятор могут являться будущим решением.

LORKA: a PASCAL-nak egy kiterjesztése relációs adat-bázis-kezelő rendszerek számára

M. Katrib Mora, E. Quesada Orozco, J. Bosch Bayard,

E. Aubert Vazquez

Összefoglaló

A LORKA egy relációs adat-báziskezelő rendszer, amely a PASCAL nyelvnek egy moduláris kibővitésén alapszik. A kibővités során uj tipusok, változók, eljárások és függvények jöttek létre. A kibővités egy olyan interfejsz--nek tekinthető, amely segitségével jól lehet dolgozni a relációs adat-bázis fogalmaival anélkül, hogy a felhasználót a rendszer müködésének részleteivel terhelné.

LORKA: an extension of PASCAL for the relational data-base models

M. Katrib Mora, E. Quesada Orozco, J. Bosch Bayard,

E. Aubert Vazquez

Summary

LORKA is a system for treating the relational data-base models that is based on a modular extension of PASCAL. In the course of extension new types, variables, procedures and function have been created. The extension can be considered as a good interface enabling good work with the basic notions of relational data-base models but at the same time the user is not burdened by the details of the work of the system itself.