Методы реализации сетевых структур посредством реляционных СУБД

Поскольку, с одной стороны, базы ресурсов СИЛТ являются специальной физической моделью реляционной СУБД и, с другой стороны, использование в СППР деревьев решений предполагает наличие определённого сценария (алгоритма), в процессе выполнения которого, перемещаясь от одного узла к другому, выполняется поиск решения, то будет целесообразным рассмотреть методы реализации сетевых структур посредством реляционных СУБД

Анализ методов представления сетевых структур

Нелинейные информационные структуры

Поскольку сетевые структуры относятся к классу нелинейных информационных структур, то напомним относящиеся к ним основные положения [Дональд Кнут. Искусство программирования].

Дерево формально определяют как конечное множество T, состоящее из одного или более узлов, таких, что: имеется один, специально обозначенный узел, называемый корнем данного дерева. Остальные узлы, исключая корень, содержатся в $n \geq 0$ попарно непересекающихся множествах T_1, T_2, …,T_n, каждое из которых в свою очередь является деревом. Деревья T_1, T_2, …,T_n называются поддеревьями данного дерева. Определение рекурсивно, поэтому каждый узел дерева является корнем некоторого поддерева, которое содержится в этом дереве. Число поддеревьев данного узла называют степенью этого узла, а узел с нулевое степенью называют концевым узлом, листом или терминальной вершиной. Уровень узла по отношению к дереву T определяется следующим образом: говорят, что корень имеет уровень 1, а другие узлы имеют уровень на единицу выше уровня своего корня. На рис. ?? и ?? изображены деревья с семью узлами. Корнем дерева является узел A.

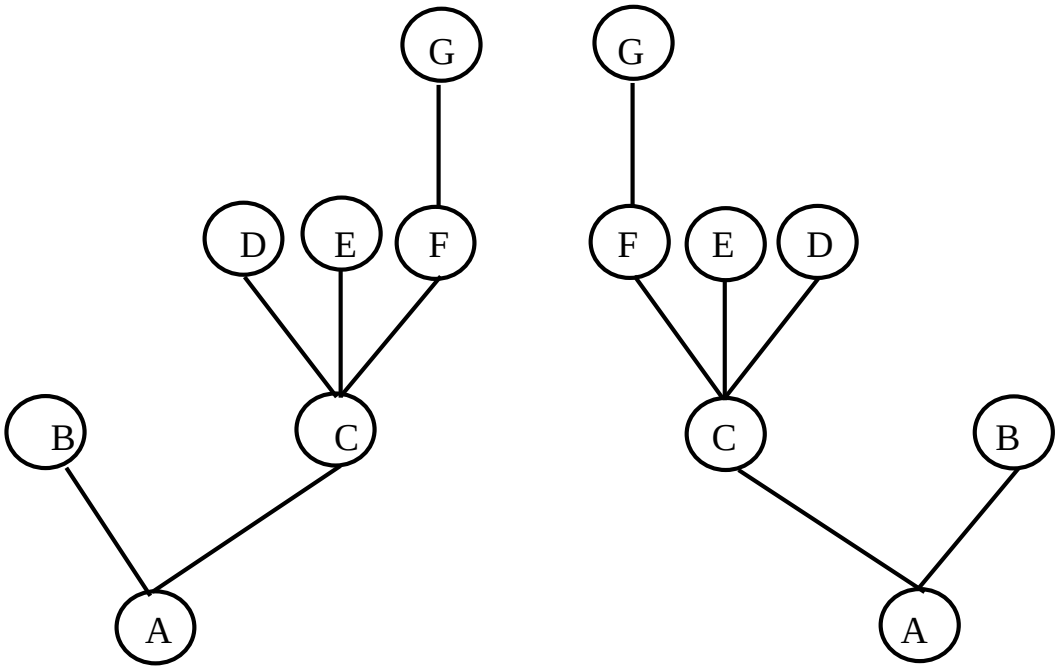


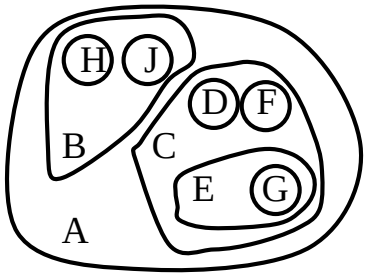
Рис. 114. Пример дерева Рис.115. Пример симметричного дерева

Дерево на рис. 114 имеет поддеревья {B} и {C, D, E, F, G}. Корнем дерева {C, D, E, F, G} является узел C, имеющий уровень 2 относительно всего дерева. В свою очередь, он имеет три поддерева: {D}, {E}, {F,G} и, следовательно, его степень равна 3. Узлы B, D, E, G являются концевыми узлами. F – единственный узел степени 1, а G – единственный узел уровня 4.

Подразумевается, что все рассматриваемые нами деревья являются упорядоченными, так что деревья, изображённые на рис. 114 и 115 считаются различными.

Лес – это множество (обычно упорядоченное), состоящее из некоторого числа непересекающихся деревьев.

Деревья могут изображаться другими структурами или, наоборот, служить изображением других логических структур, например, вложенных множеств, вложенных списков, вложенных уступов и т.п. (рис. 116).



(A (B (H) (J))(C (D) (E (G)) (F)))

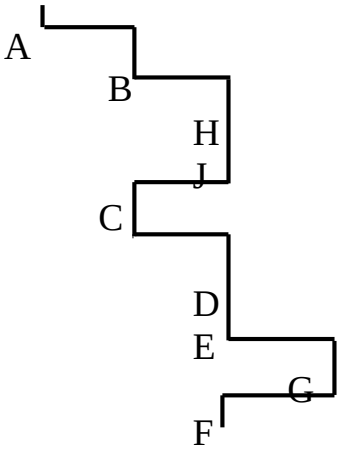


Рис. 116. Примеры древовидных структур

Так всякий прямоугольный массив можно рассматривать как частный случай древовидной структуры. Например, вот два представления матрицы размера 3*4 (Рис. 117):

$$\begin{pmatrix} A[1,1] & A[1,2] & A[1,3] & A[1,4] \\ A[2,1] & A[2,2] & A[2,3] & A[2,4] \\ A[3,1] & A[3,2] & A[3,3] & A[3,4] \end{pmatrix}$$

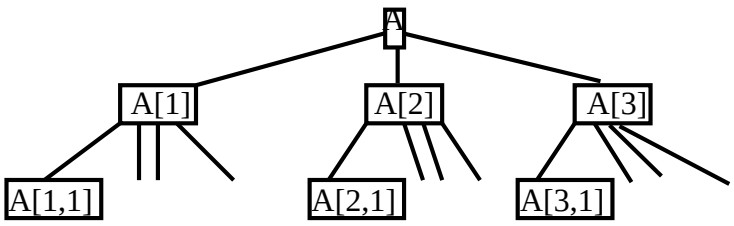


Рис. 117. Представление массива в виде дерева

Другой пример древовидной структуры дают нам алгебраические формулы. Например, формулу *a-b*(c/d+e/f)* можно изобразить в соответствии с рис. 118.

Произвольные деревья представляются посредством бинарных деревьев. Бинарное дерево определяется как конечное множество узлов, которое или пусто или состоит из корня или из двух непересекающихся поддеревьев, называемых правым и левым поддеревьями данного корня.

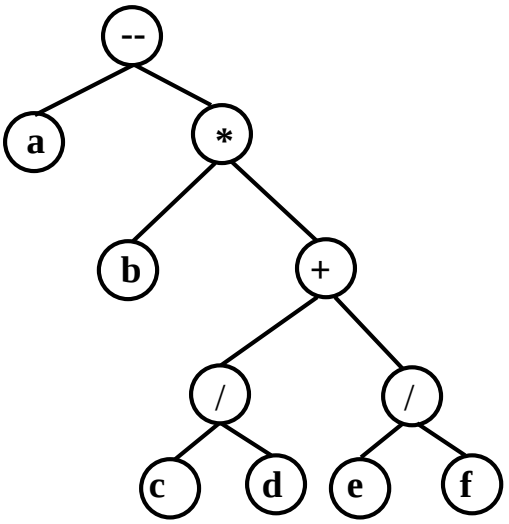


Рис. 118. Пример представления алгебраического выражения в виде дерева

Использование бинарных деревьев удобно в силу их регулярности – каждый узел имеет степень не более двух. Переход от произвольного леса к бинарному дереву выполняют на основе следующего правила (рис. 119):

- соединяют между собой корневые вершины;
- соединяют между собой сыновей каждой семьи;
- убирают все связи, идущие от родительской вершины к сыновьям, кроме связи, между родительской вершиной и первым сыном.

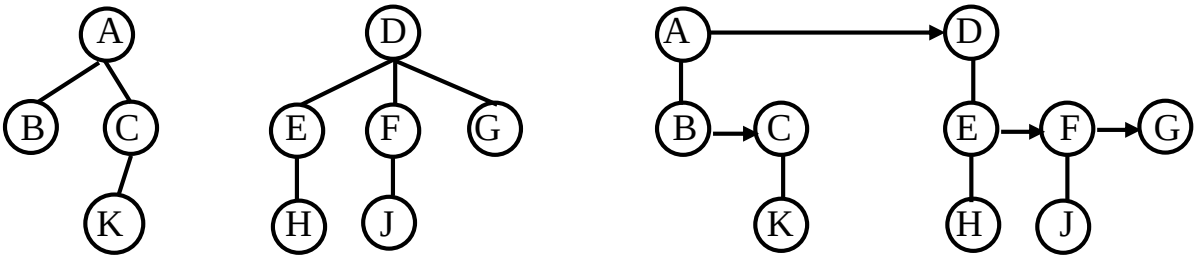


Рис. 119. Иллюстрация перехода к двоичному дереву

Одним из естественных способов представления бинарных деревьев в ЭВМ является представление в виде записей, содержащих три поля (рис. 120):

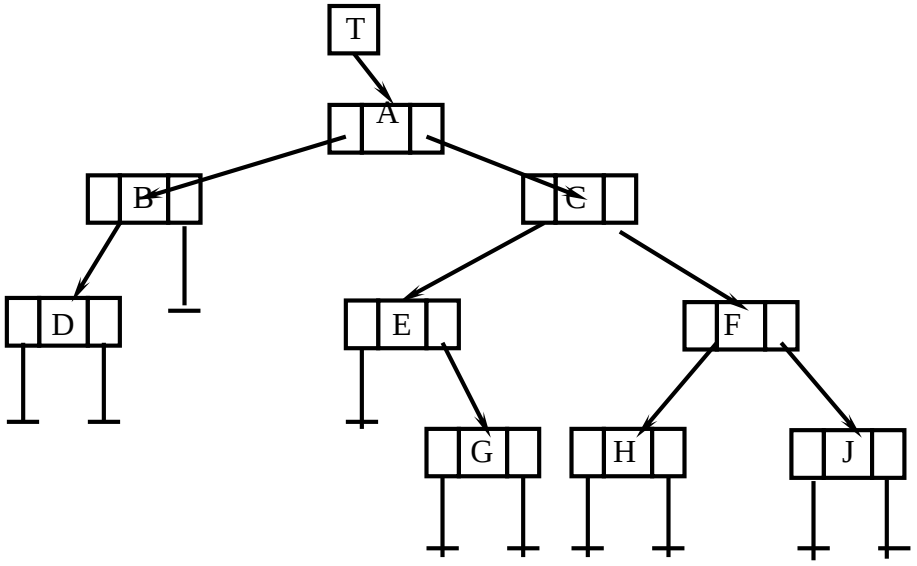


Рис. 120. Иллюстрация представления дерева в виде двухсвязного списка

- Key -- ключевое или информационное поле, отождествляемое с узлом;
- Llink -- указатель на левое поддерево;
- Rlink -- указатель на правое поддерево;

Помимо метода Llink-Rlink (левый сын-правый брат), Существует еще много способов программной реализации древовидных структур. Обычно надлежащий выбор представления в большей степени определяется видом операций над деревьями, которые предстоит выполнить.

В процессе работы с деревьями необходимо уметь выполнять следующие основные операции:

- введение новой вершины.
- удаление указанной вершины.
- навигация по дереву.
- обход или прохождение дерева. Это способ методичного исследования узлов дерева, при котором каждый узел проходится точно один раз. Полное прохождение дерева даёт нам линейную расстановку узлов.

Для прохождения бинарного дерева чаще всего используют один из двух способов: проходить узлы в прямом или обратном порядке. Прямой порядок определяют следующим правилом:

1. Попасть в корень.
2. Пройти левое поддерево.
3. Пройти правое поддерево.

Обратному порядку соответствует правило:

1. Пройти левое поддерево.
2. Попасть в корень.
3. Пройти правое поддерево.

Учитывая, что базы ресурсов СИЛТ хранятся в реляционных базах данных, из множества методов представления ориентированных сетевых структур (без циклов) средствами реляционных СУБД рассмотрим четыре метода:

- иерархический;
- избыточный двухсвязный список;
- трёхсвязный список;
- на основе классификации Дьюи.

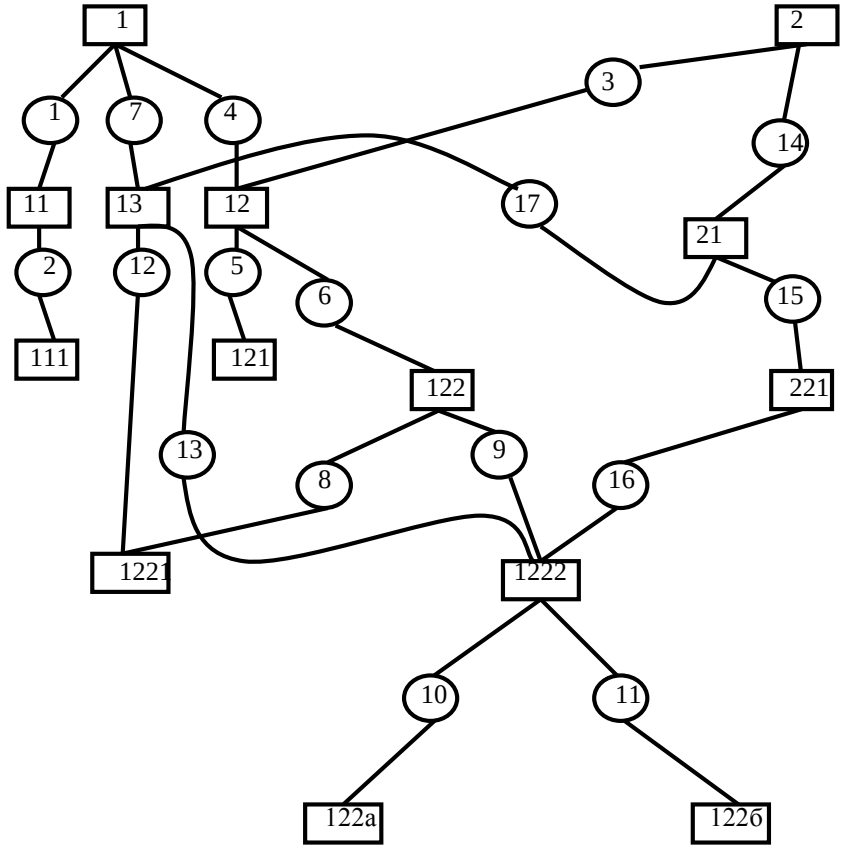


Рис. 121. Фрагмент сетевой структуры

Анализ этих методов выполним на основе примера (рис. 121)

Иерархическое представление ориентированной сети без циклов

Для такого представления достаточно всего два поля Id и Parent. Создадим таблицу (таблица 14) и заполним на основе сети, изображённой на рис. 121, поля Id и Parent.

Таблица 14

Номер	Id	Parent
1	11	1
2	111	11
3	12	2
4	12	1
5	121	12
6	122	12
7	13	1
8	1221	122
9	1222	122
10	122а	1222
11	122б	1222

12	1221	13
13	1222	13
14	21	2
15	221	21
16	1222	221
17	13	21

В результате мы получим неупорядоченный набор иерархических отношений. Задача в такой постановке возникает при создании базы данных спецификаций изделий в масштабе предприятия и решения задач входимости, применяемости изделий, формирования ведомости материалов, создания комплектовочных карт изделий и целого ряда других подобных задач.

Также следует обратить внимание, что в точно такой же постановке можно сформулировать задачу построения сети вывода над базой правил ЕСЛИ-ТО, в предположении, что содержимое условия «ЕСЛИ» ассоциируется с узлом Id, а содержимое заключения «ТО» - с узлом «Parent».

Такое представление является достаточным для решения большинства задач, связанных с навигацией по дереву, однако оно требует повышенных расходов вычислительных ресурсов, из-за многократного поиска записи в процессе перехода к последующему узлу. Более удобным является использование представления, содержащее дополнительные поля LLINK и RLINK.

Метод представления сетевых структур без циклов на основе избыточного двухсвязного списка.

Такой метод требует одно ключевое поле key, а также поля для указателей LLINK и RLINK. Однако для повышения степени инвариантности представления, а также наглядности и удобства изложения последующих методов, в качестве информационного поля мы будем использовать ключевое поле **Id** и в качестве родительского поле **Parent**. Ключом в этом случае может являться конкатенация этих полей. Однако для повышения степени инвариантности представления добавим суррогатный первичный ключ «Номер записи». Для иллюстрации метода создадим нижеследующую таблицу (таблица 15). В соответствии с выше приведенном на рис. 121 фрагментом сети, заполним поля Id, Parent, Llink и Rlink, принимая во внимание, что каждая пара Id-Parent определяется на рисунке верхним и нижним прямоугольником, связанным линией с кружком. Номер в нижнем прямоугольнике заносится в поле Id. Номер, помещенный в кружке линии, связывающей два прямоугольника, является суррогатным первичным ключом записи таблицы, представляющей пару Id-Parent.

При заполнении полей Llink и Rlink следует учитывать следующие правила:

- в поле Llink заносится ключ, соответствующей старшему сыну семьи. Если сыновья отсутствуют, то записывается -1;
- в поле Rlink заносится номер записи, соответствующей брату данной вершины. Если вершина является крайней в семье, то в это поле заносится -1 .

В результате мы получим представление сетевой структуры в виде избыточного двухсвязного списка.

На рис. 122 приведён алгоритм прямого обхода с последующими комментариями к нему, который может быть реализован на любом из алгоритмических языков программирования.

Для программного формирования такого представления на основе иерархического представления можно использовать алгоритмы, приведённые на рис. 124 - 125.

В качестве контрольного результата правильности работы этого алгоритма могут служить значения стека ключей, которые должны содержать значения узлов сети, представляемых значениями поля Id в порядке прямого обхода.

Сделаем следующие замечания:

- рассматриваемый пример иллюстрирует возможность работы с сетевой структурой, поскольку повторяющиеся поддеревья записываются в таблицу один раз;
- алгоритм использует для навигации по базе данных только команду перехода по первичному ключу. Этот способ является наиболее быстрым во всех СУБД;
- алгоритм может быть реализован также и на основе структуры, содержащейся в оперативной памяти;

Таблица 15. Пример реляционной таблицы, содержащей фрагмент ориентированной сети без циклов

Номер	Id	Parent	LLINK	RLINK
1	11	1	2	4
2	111	11	-1	-1
3	12	2	5	14
4	12	1	5	7
5	121	12	-1	6
6	122	12	8	-1
7	13	1	12	-1
8	1221	122	-1	9
9	1222	122	10	-1
10	122а	1222	-1	11
11	122б	1222	-1	-1
12	1221	13	-1	13
13	1222	13	10	-1
14	21	2	17	-1
15	221	21	16	-1
16	1222	221	10	-1
17	13	21	12	15

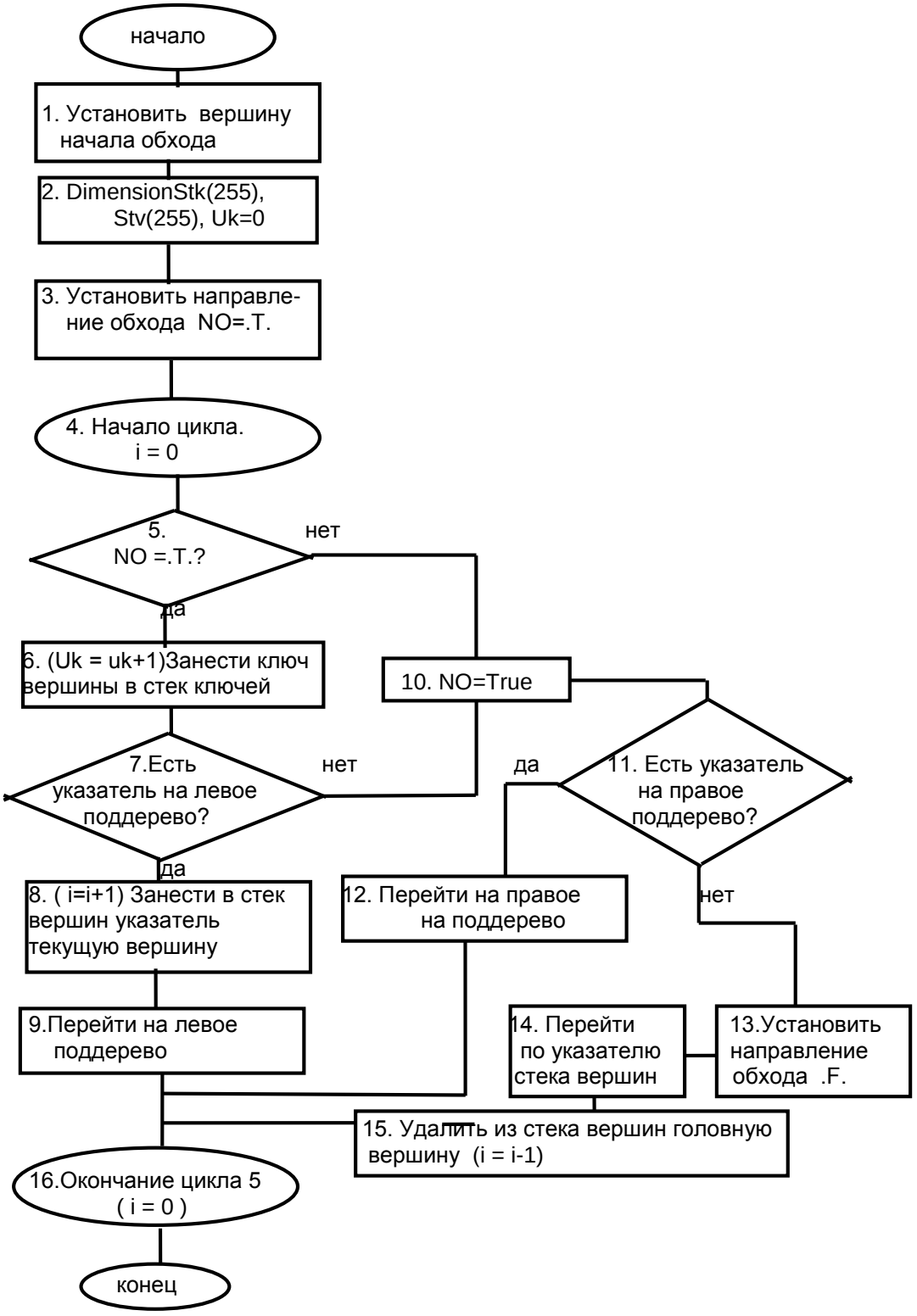


Рис. 123. Алгоритм прямого (левостороннего) обхода дерева

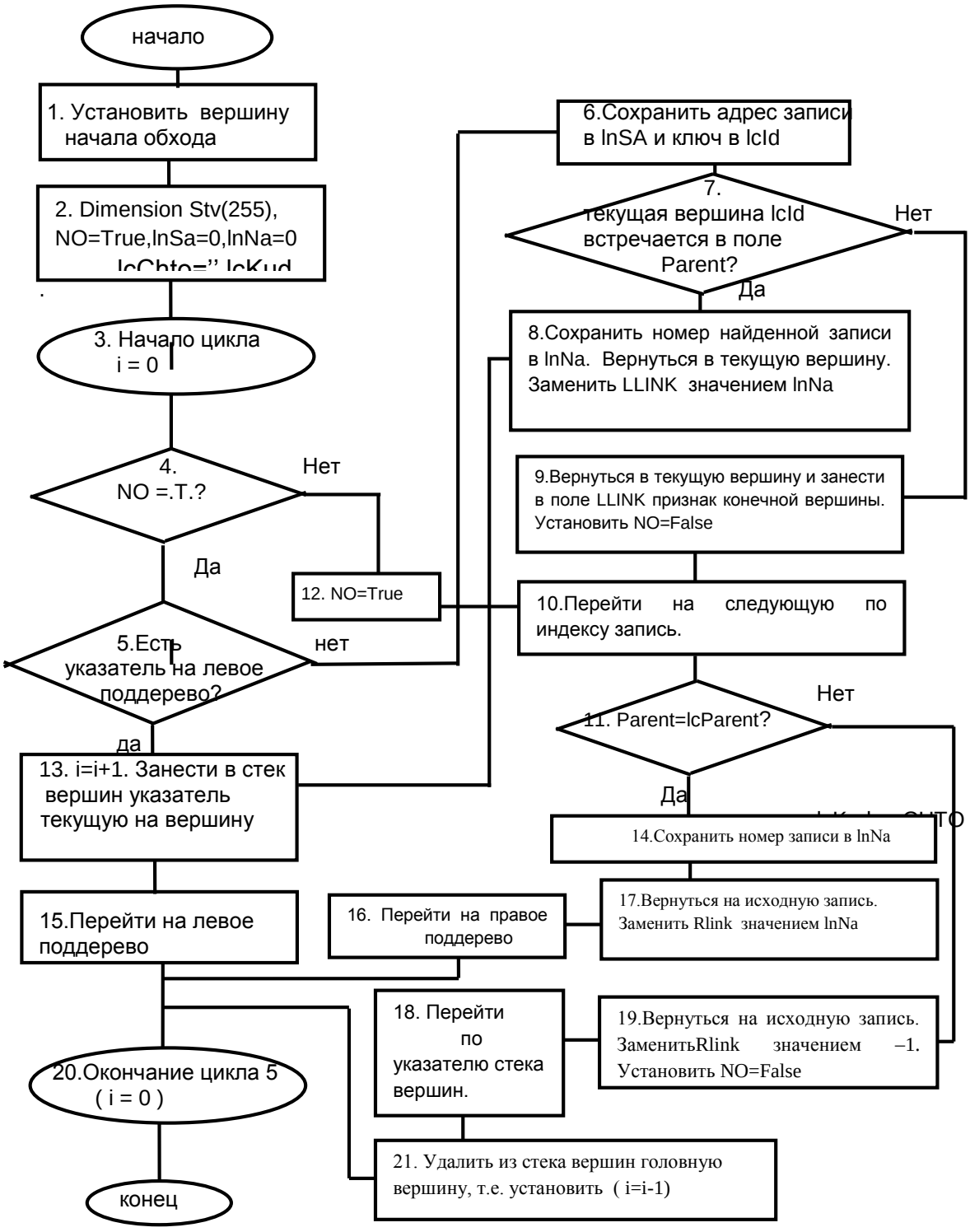


Рис. 124. Алгоритм процедуры ProshPod построения бинарного поддерева.

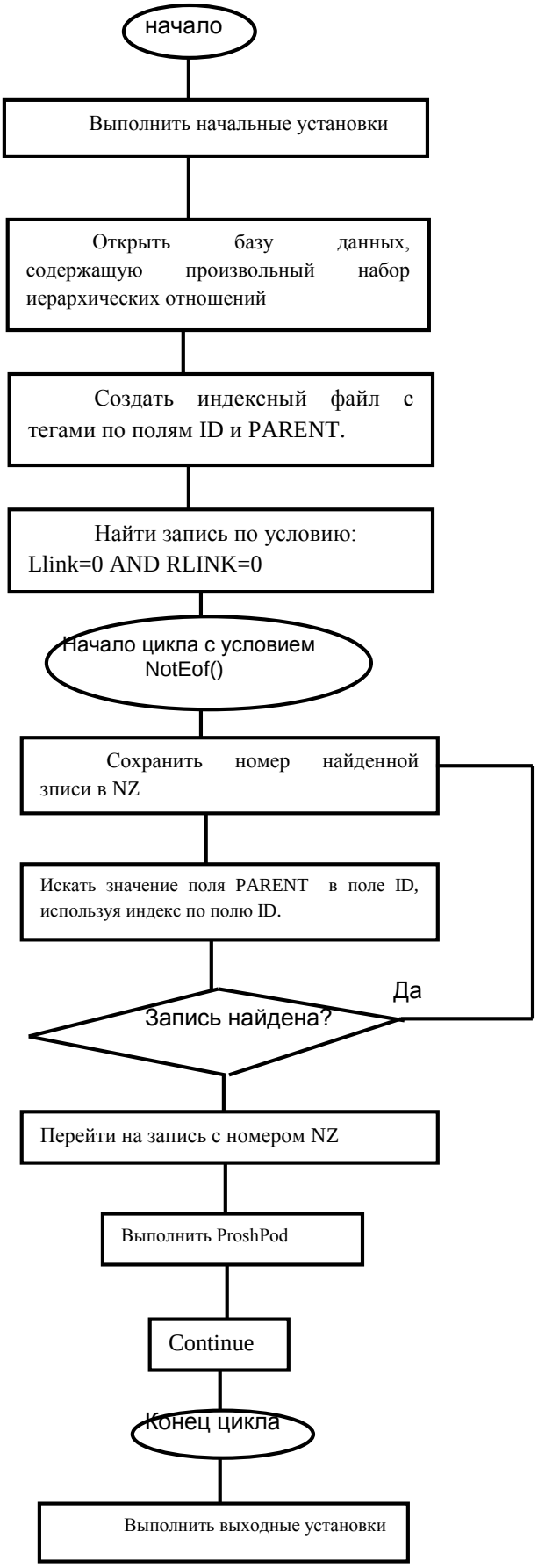


Рис. 125. Алгоритм построения двоичного дерева над неупорядоченным “лесом” деревьев

Комментарии к алгоритму прямого (левостороннего) обхода дерева (рис. 123)

Оператор 1. Установление вершины начала обхода. В качестве начала обхода может быть принята любая вершина сети. Начало может быть установлено путем передачи в процедуру номера записи, с которой начинается обход.

Оператор 2. Инициализируются массивы, выполняющие роль стеков имен вершин и ключей вершин, т.е. ассоциированных с вершинами данных. Кроме того инициализируются целые переменные, выполняющие роли указателей вершин стеков.

Оператор 3. Установить направление обхода **True.** Предполагается инициализация логической переменной, которой присваивается значение «истина» и которое указывает на направление обхода от корня к листьям.

Оператор 4. Начало цикла. (Имеется в виду команда определения цикла DO WHILE)

Оператор 6. Занести имя вершины в стекимён вершин. Это действие выполняет роль процедуры обработки узла дерева при прямом обходе. В настоящий момент для упрощения иллюстрации метода в качестве имени удобно принять значение поля Id.

Оператор 7. Есть указатель на левое поддерево? Означает проверку возможности перехода к вершине-сыну по отношению к текущей вершине.

Оператор 8. Занести в стек вершин указатель на текущую вершину. Означает, что необходимо увеличить указатель головы стека на 1 и занести номер записи текущей вершины в массив, выполняющий роль стека.

Оператор 9. Перейти на левое поддерево. Означает переход на запись, соответствующую вершине-сыну относительно текущей вершины.

Оператор 10. Установить направление обхода True. Т.е. восстановить первоначальное направление обхода.

Оператор 11. Означает проверку возможности перехода на правое поддерево, которое определяется положительным указателем RLINK на правое поддерево.

Оператор 12. Перейти на правое поддерево. Означает переход на запись, соответствующую вершине-брату относительно текущей вершины. Здесь следует напомнить, что речь идет о двоичном представлении произвольного дерева. В таком представлении на общепринятом жаргоне может идти речь о вершине-отце, вершине-сыне и вершине-брате. Других вершин в двоичном представлении нет.

Оператор 13. Установить направление обхода False. Изменяет первоначальное направление обхода на обратное.

Оператор 14. Перейти по указателю стека вершин. Это действие означает чтение головы стека вершин и перехода по выбранному из него указателю на родительскую вершину относительно текущей вершины.

Оператор 15. Удалить из стека вершин головную вершину. Означает просто уменьшение указателя стека вершин на 1.

Оператор 16. Окончание цикла (возврат на начало цикла).

Избыточный двухсвязный список можно легко преобразовать в строгий двухсвязный список простым удалением поля Parent, или в односвязный – удалением поля Rlink и использованием индекса по полю Parent.

Еще одну разновидность представления можно получить, оставив поля Id и Llink и введя поле уровня вершины.

И наконец, вариант представления, который удобно использовать, в основном, для визуализации фрагментов древовидных структур – это упорядоченная в порядке левостороннего (прямого) обхода последовательность ключей узлов и соответствующая последовательность уровней каждого узла.

Представление ориентированных сетевых структур без циклов на основе трёхсвязного списка.

Рассмотрим представление сетевых структур, обеспечивающее наиболее быструю обработку и наибольшую степень инвариантности. Оно представляет собой модификацию избыточного двухсвязного списка следующим образом:

- ключевое поле Parent преобразовывается в указатель на родительскую вершину;
- указатель Llink становится указателем Next на следующую вершину в порядке левостороннего обхода;
- указатель Rlink остаётся без изменения и указывает всегда на правое поддерево, если оно существует. В противном случае Rlink содержит -1;
- для ускорения выполнения обратных обходов для сетевых структур вводится логическое поле Rliki, которое принимает значение True, если узел имеет более одной родительской вершины;
- для использования этого представления в качестве сети вывода добавляется еще поле для представления количественной характеристики, интерпретируемой в различных контекстах различным образом: степень достоверности в сетях логического вывода; весовая характеристика при обучении нейронной сети и т.п.

Представление древовидных структур на основе классификации Дьюи.

Еще одним из способов представления деревьев в реляционной базе данных, является такое представление, при котором каждая вершина имеет своё уникальное обозначение, которое получается путем добавления к обозначению родительского узла собственного уникального номера. Такой метод часто называют системой обозначений Дьюи для деревьев, по аналогии с классификационной схемой, применяемой в библиотеках. Пример такой реализации представлен на рис. 126.

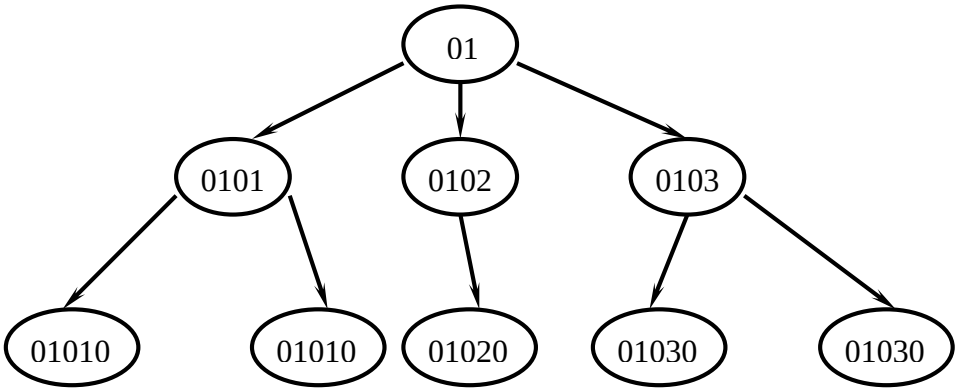


Рис. 126. Представление дерева в форме классификации Дьюи

Десятичная система Дьюи применима ко всякому лесу: корень k-того дерева в лесу обозначается номером k; если α-номер любого узла степени m, то его сыновья

обозначаются номерами α.1, α.2, …, α.m. Вместо десятичной системы счисления в реальных приложениях употребляют системы с основанием 128 или 256 и отводят более чем одну позицию для нумерации сыновей любого узла. Так например, в случае системы с основанием 256 и двухбайтовой позиции получим возможность оперировать более чем с 65000 членов семьи одного узла.

Обратим внимание на следующие важные свойства представления деревьев в форме классификации Дьюи:

- обозначения узлов представляют собой строки символов;
- узел, имеющий уровень k имеет обозначение длиной $k*n$ символов, где n – число позиций, отведённое под номер;
- последние n символов представляют порядковый номер узла в составе семьи;
- для любого узла первые $k*n-n$ символов являются обозначением родительской вершины.

Система обозначений Дьюи имеет много простых математических свойств и является полезным инструментом при анализе деревьев.

Для представления “леса” на основе классификации Дьюи необходима таблица базы данных (например Tree), содержащая, как минимум, два поля:

- в поле Tree.NomVer хранится индекс вершины дерева;
- в поле Tree.Link – ключ записи таблицы, содержащей данные, ассоциированные с данным узлом.

Для такой таблицы создаются обычно три индекса:

- индекс по полю NomVer;
- индекс на основе ключа Str(len(trim(NomVer))) + NomVer;
- индекс по полю Link.

Индекс по полю NomVer обеспечивает упорядочивание индексов вершин в соответствии с прямым левосторонним обходом.

Индекс на основе ключа Str(len(trim(NomVer))) + NomVer обеспечивает упорядочивание в соответствии с уровнями дерева, т. е. сначала будут идти вершины первого уровня, затем – второго и т.д.

Индекс по полю Link обеспечивает прямой доступ к узлам дерева по ассоциированным с узлами данным и, одновременно, группирование одинаковых данных, но встречающихся в различных контекстах.

Главным недостатком представления деревьев посредством классификации Дьюи является невозможность представления ориентированных сетей без циклов, не повторяя общие поддеревья.

Выводы по разделу «Методы реализации сетевых структур в реляционных СУБД»

Рассмотренные выше варианты представления сетевых структур средствами реляционных СУБД позволяет решить три задачи:

- создать вычислительную среду для естественного манипулирования тем большим числом различных классификаторов объектов СППР, которые используются в инженерной практике, т.е. создать реализацию классификационной компаненты (см. выше раздел «Классификационная компанента»;

- создать вычислительную среду для однозначного формирования имён атрибутов, относящихся к различным объектам СППР;
- обеспечить методическую базу для решения задач искусственного интеллекта, основанных на применении сетевых моделей;
- добавить методическую базу для реализации ИЛТ форме прямого и обратного дерева вывода.