МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

РЕФЕРАТ НА ТЕМУ:

«Логические языки и базы данных»

Студент группы 8О-206Б Касимов М.М.

Преподаватель: Сошников Д.В.

Реферат сдан « » г.

Оценка:

Подпись преподавателя:

ОГЛАВЛЕНИЕ:

1. Введение
2. Языки логического программирования
3. Prolog
4. Реализация баз данных
5. Заключение
6. Список использованной литературы

**Введение**

Базы данных использовались в вычислительной технике с незапамятных времен. В первых компьютерах использовались два вида внешних устройств – магнитные ленты и магнитные барабаны.

Системы баз данных сегодня являются основой построения большинства информационных систем и используются при автоматизации практически всех сфер человеческой деятельности. Например, доступ к базе данных необходим при работе с библиотечной информационной системой, содержащей сведения обо всех книгах, имеющихся в библиотеке, ее читателях, заявках на бронирование книг и т.д. В ней обычно содержатся средства, позволяющие читателям находить нужную им книгу по названию, фамилиям авторов или указанной тематике. С помощью такого рода систем организуется учет движения книг, другие операции, необходимые в библиотечной деятельности. В ВУЗе могут существовать базы данных с информацией о студентах, профессорско-преподавательском составе, факультетах и кафедрах, др. данные, необходимые для функционирования так называемых комплексных информационно-аналитических систем и их подсистем (учета кадров, бухгалтерской, документооборота, информационного обеспечения учебной деятельности и т.п.). Базы данных по народонаселению содержат сведения о жителях города, региона и т.п., необходимые для функционирования систем налогообложения, здравоохранения, образования, социальной защиты, др.

Людям хотелось облегчить работу с базами данных, так и появились инструменты и языки программирования, направленные на работу с ними.

К таким инструментам относятся: SQL, Python, Scala. Но тем не менее, с базами данных, так же справляются и логические языки программирования. Главный их плюс – это их удобство для человека. Людям сложно работать только с процедурной семантикой, а у логических языков программирования хорошо развита декларативная семантика. К тому же, в них хорошо реализованы готовые функции для работы с базами данных, например, findall, о котором мы еще поговорим.

**Языки логического программирования**

Логика как направление знаний появилась более 2000 лет назад в Древней Греции. Основная цель логики – формализация человеческих способов рассуждения, то есть сведение мыслительной деятельности человека к некоторым действиям с цифрами и символами.

Стиль программирования, при котором решение и алгоритм решения создается компьютером автоматически, а для решения необходимо формальное описание, называют декларативным, а соответствующие языки – декларативными языками программирования.

Алгоритмической моделью, лежащей в основе логического программирования, можно считать логику предикатов, которая является частью дискретной математики. Можно считать, что логической программой является описание соответствующей задачи на языке логики предикатов, а собственно выполнение инициируется некоторым запросом на нахождение значений переменных, удовлетворяющих какому-либо предикату или формуле.

Компьютеры пока не в состоянии встать в один ряд с человеком по уровню интеллектуальной деятельности. А потому людям все еще приходится работать с процедурной семантикой, ведь компьютеры сейчас не имеют такой автоматизации, какой бы нам хотелось.

На данный момент существует немало логических языков программирования: Planner, Prolog, Mercury и т.д.

**Prolog**

Начало истории языка относится к 1970-м годам. Будучи декларативным языком программирования, Пролог воспринимает в качестве программы некоторое описание задачи или баз знаний и сам производит логический вывод, а также поиск решения для задачи, пользуясь механизмом поиска с возвратом и унификацией.

Интерес к Прологу поднимался и затихал несколько раз, энтузиазм сменялся жёстким неприятием. Наиболее высоко был поднят интерес к языку Пролог, как к языку будущего, во время разработок японской национальной программы компьютеры пятого поколения в 1980-х годах, когда разработчики надеялись, что с помощью Пролога можно будет сформулировать новые принципы, которые приведут к созданию компьютеров более высокого уровня интеллекта.

Язык Пролог в 1980-х годах был включён в ряд советских вузовских и школьных учебников информатики для изучения элементов математической логики, принципов логического программирования и проектирования баз знаний и моделей экспертных систем. С этой целью на IBM PC и ряде советских школьных компьютеров были реализованы учебные русскоязычные интерпретаторы Пролога.

В языке Пролог факты описываются в форме логических предикатов с конкретными значениями. Правила вывода описываются логическими предикатами с определением правил логического вывода в виде списка предикатов над базами знаний и процедурами обработки информации.

В настоящее время Пролог, несмотря на неоднократные пессимистические прогнозы, продолжает развиваться в разных странах и вбирает в себя новые технологии и концепции, а также парадигмы императивного программирования.

Пролог реализован практически для всех известных операционных систем и платформ (в том числе для Java и .NET). В число операционных систем входят: ОС для мейнфреймов, всё семейство Unix, Windows, ОС для мобильных платформ.

Программа включает в себя множество описаний фактов и правил, содержащих такие объекты, как переменные, константы, логические связки, функциональные символы и т.д. основными смысловыми объектами программы (за исключением связок и малозначимых синтаксических элементов) являются термы.

**Атомы**. Представляют собой константы, соответствующие объектам предметной области, описываемой программой. Атомы, задаваемые одним и тем же идентификатором, на протяжении всей программы обозначают один и тот же объект:

Синтаксически, атом может представляться:

* Последовательностью символов, начинающейся со строчной буквы
* Последовательностью спецсимволов
* Произвольной последовательностью символов, заключенной в единичные кавычки

**Числа**. По своей сути числа мало отличаются от атомов, но они могут входить в арифметические выражения, для вычисления которых в языке предусмотрены встроенные предикаты. Большинство реализации Пролога поддерживают целые и действительные числа. Для того чтобы выяснить, каковы диапазоны и точность, чисел следует обратиться к руководству по конкретной реализации.

**Переменные**. Записываются в виде идентификаторов, начинающихся с заглавной буквы или подчеркивания. Область действия переменной – одно правило, в котором она встречается, соответственно две одинаковые переменные, встречающиеся в разных правилах могут обозначать различные объекты. На самом деле, система при интерпретации входной программы как правило не сохраняет информацию об именах переменных, а заменяет их внутренними ссылками.

Следует упомянуть анонимную переменную, обозначаемую знаком подчеркивания\_: каждое вхождение этой переменной по сути дела означает новую переменную. В каждый момент времени переменная может быть свободной и связанной. Изначально переменные в определении предикатов подразумеваются свободными. В процессе доказательства производится унификация, в результате которой переменные унифицируются с константами или другими переменными – связываются. Повторная унификация со связанной переменной не позволяет изменить ее значение, а лишь произвести его проверку на соответствие некоторой константе или другому значению. Значение переменной может измениться только при возврате назад, когда система поднимается вверх по дереву SLD-резолюции и начинает рассматривать другую ветвь решений, на которой данная переменная унифицирована с другим значением.

**Структурный терм**. Конструкция вида func(t1,…,tn), где func – функциональный символ или функтор, а t1,…,tn – другие термы, простые или структурные. При этом говорят, что функтор имеет арность n. Структурные термы удобно использовать для представления сложных структур данных, среди которых есть списки. Списки в Пролог играют большую роль.

При доказательстве логической программы методом резолюции фундаментальным методом является унификация. Правила унификации:

* Свободная переменная унифицируется с произвольным термом (в том числе с другой переменной, свободной или связанной) и связывается;
* Связанная переменная унифицируется как значение, с которым она связана;
* Константа унифицируется только с такой же константой; две отличные друг от друга константы не унифицируются
* Структурные термы унифицируются в том случае, если у них одинаковый функтор и арность, и все их аргументы попарно унифицируются

Нотация структурных термов, принятая в логике предикатов, по сути дела представляет собой префиксную нотацию, в которой сначала записывается функтор-оператор, а затем – аргументы. В то время как такая запись удобна для многих функций, в математике в ряде случаев используется инфиксная и постфиксная нотация, когда функтор записывается между аргументов либо после них соответственно.

Пролог позволяет пользователю описывать дополнительные унарные или бинарные операторы, использую для этого префиксную, инфиксную или постфиксную нотацию. Для объявления оператора используется встроенный системный предикат op, имеющий следующий формат вызова:

:-op(<приоритет>,<шаблон>,<оператор>).

* Приоритет – некоторое число, используемое системной для определения оператора относительно других операторов. Как правило, большее значение этого параметра соответствует более низкому приоритету оператора.
* Шаблон задает одновременно арность (унарный или бинарный), позиционность (префиксный, инфиксный, постфиксный) и ассоциативность (левоассоциативный, правоассоциативный, неассоциативный) оператора.
* Оператор может быть последовательностью спецсимволов либо функциональным символом

Любая система программирования, в том числе на логических языках, как правило содержит некоторое количество системных функций для выполнения операций, связанных с программным окружением. В частности, к таким функциям относится ввод-вывод.

Понятие ввода-вывода является основополагающим для императивных языков, поскольку позволяет разрабатывать массовые алгоритмы по обработке информации, поступающей на вход программе, с выдачей результатов на выходе. Для классических языков логического программирования запросным режимом работы необходимость в явных операциях ввода-вывода не возникает, поскольку его функции берет на себя интерпретатор запросов. Тем не менее, возможность ввода-вывода может оказаться полезной, а в системах программирования, не имеющих диалогового режима – необходимой. Например, write, позволяет вывести что-то на консоль.

Сегодня язык программирования Prolog не ограничивается лишь использованием научно-исследовательских лабораториях, а иногда рассматривается как мощный инструмент для разработки практических приложений.

Пролог критикуется, в первую очередь, за неполную декларативную природу: создание сколько-нибудь сложных и практически полезных Пролог-программ в полностью декларативном стиле практически невозможно, программист вынужден прибегать к процедурным приёмам, что приводит к резкому возрастанию сложности создания и отладки программ, а также плохой контролируемости промежуточных результатов.

**Реализация базы данных.**

**Дедуктивная база данных** является системой базы данных, которая может сделать выводы (то есть, заключать дополнительные факты) на основе правил и фактах, хранящихся в (дедуктивных) баз данных.

Слияние логических языков программирования и баз данных произошло благодаря тенденции совместно использовать различные области в информатике для получения новых идей и получение выигрыша за счет использования общих концепций. Успеху интеграции способствовало то обстоятельство, что парадигма логического программирования, была выбрана в качестве главной опоры в японском проекте ЭВМ пятого поколения.

Логическое программирование и управление базами данных имеют общие черты:

1. Базы данных. Системы логического программирования управляют небольшими базами данных, располагаемыми в оперативной памяти. Системы баз данных, напротив, управляют большими, находящимися в коллективном пользовании совокупностями данных в массовой памяти.
2. Запросы. В том и другом есть запросы, с помощью которого извлекается та или иная информация.
3. Ограничения. То есть они оба задают условия правильности баз данных.

Естественное расширение логического программирования и управления базами данных заключается новых классов систем. Такие системы основаны на использовании логического программирования в качестве языка запросов и объединяют формулирование запросов и ограничений в стиле логического программирования с технологией баз данных для эффективного и безопасного запоминания данных в массовой памяти.

**Связывание** системы логического программирования с реляционной системой баз данных состоит в определении архитектуры связи двух систем для того, чтобы позволить высокоуровневые запросы и эффективное манипулирование данными. Реализацию системы, объединяющую логическое программирование и систему управления баз данных, можно разделить на следующие четыре метода:

1. Связывание существующего языка логического программирования с существующей системой управления базами данных.

2. Расширение существующего логического языка путем добавления некоторых способностей реляционных систем баз данных.

3. Расширение существующей реляционной системы баз данных путем добавления некоторых свойств языка логического программирования.

4. Плотная интеграция технических приемов логического программирования с приемами реляционных систем управления баз данных.

На сегодняшний день больше всего рассматривается именно первый способ, так как дедуктивная система может извлечь пользу из будущих независимых друг от друга развитий систем логического программирования и систем управления базами данных. Уровень связывания между двумя этими системами можно разделить на два вида:

1. Слабое связывание. В таких системах взаимодействие между системой логического программирования и системой управления базами данных происходит во время загрузки, то есть система логического программирования определяет неинициализированные предикаты базы данных, выдает соответствующие запросы базе данных и добавляет результат как факт программы. Во время выполнения программы на языке логического программирования запросы базе данных не отправляются

2. Сильное связывание. В таких системах система логического программирования и система управления базами данных взаимодействуют во время процесса вывода. В более простых системах каждый раз, когда система логического программирования пытается удовлетворить предикатам из базы данных, к базе данных отправляется запрос. В более сложных системах система логического программирования находит успешные предикаты базы данных и сравнения (базовые конъюнкции) и преобразовывает их вместе в один запрос, который потом отправлен к базе данных.

Слабое и сильное связывание различаются сложностью, требуемой памятью и производительностью. При слабом связывании исполняется меньше запросов к базе данных, так как каждый предикат или правило рассматривается один раз и навсегда, тогда как при сильном связывании каждое правило или предикат могут рассматриваться по нескольку раз. Однако запросы в слабо связанных системах менее селективны, чем запросы в сильно свзязанных системах.

В логическом языке Prolog есть немало предикатов для работы с базами данных: assert (для добавления фактов в базу данных), retract (для удаления фактов из базы данных), findall (для поиска) и т.д.

**Заключение**

Логические языки программирования довольно хорошо справляются с базами данных.

Что говорить, если программы для того же Пролога представляют из себя совокупность фактов и правил т.е. некую базу данных.

Логические языки программирования обладают большей выразительной способностью задания запросов и более естественной для человека декларативной семантикой при работе с базами данных в сравнении с другими языками. Более того, представление запросов возможно при этом в однородном формализме, и их проверка требует одних и тех же механизмов вывода, что позволяет вести более сложные рассуждения о содержимом базы данных. Но у логического программирования есть свои недостатки при работе с базами данных. К ним относятся: процедурность и чувствительность к порядку, покортежная обработка (нам хотелось бы, чтобы результатом запросов к базе данных было множество кортежей, но Пролог возвращает отдельные кортежи по одному.) и специальные предикаты (Программисты на Прологе управляют выполнением программ с помощью специальных предикатов, которые используются для ввода-вывода или для воздействия на бэктрекинг.)

Альтернативой Прологу как языку баз данных и логического программирования явился Дейталог, в котором были устранены указанные выше недостатки Пролога при работе с базами данных.

**Используемая литература**

1.Д.В. Сошников «Парадигма логического программирования».  
2.И. Братко «Программирование на языке Пролог для искусственного интеллекта». М: Мир 1990г.

3.Хоггер «Введение в логическое программирование». М: Мир 1988г.

4.Лекции Д.В. Сошникова по курсу «Логическое программирование» МАИ.