قـوة البرولـوغ في بناء النظم الخبيرة

هند محمد الخطبا و محمد جمال اللبن و عدنان عمورة

قسم الرياضيات ـ كلية العلوم ـ جامعة دمشق ـ سورية

تاريـخ الإيداع 2006/01/25
قبل للنشـر في 2006/05/02

الملخص

نحاول في هذا البحث إظهار المميزات الأساسية للغة البروالوغ التي تجعلها لغة تعبيرية قوية تستخدم في كتابة النظم الخبيرة وتفتقر لها العديد من اللغات التقليدية مثل لغة Pascal وغيرها. كما نقدم نظامـاً خبيراً الهدف منه ضبط المخزون Inventory Control بتطبيـق النمـوذج Fixed-Order Quantity Model. ومن خلاله وضحنا مفهوم البيانات الساكنة والبيانات الديناميكية بلغة البرولـوغ، ثـم أجرينـا مقارنة بين قواعد بيانات البروالوغ واستعلاماتها مع قواعد البيانات Access واستعلامات لغة SQL.

تعد النظم الخبيرة أحد تطبيقات الذكاء الصنعي الأساسية وتعرف غالباً بالنظم المعتمدة على القواعد. والنظم الخبيرة هي تطبيقات حاسوبية تجسد خبرات غير خوارزمية لحل أنواع محددة من المسائل. على سبيل المثال المسائل التي يمكنها أن تقدم النصيحة أو التحليـل أو التـصنيف أو التفـسير أو التعلـيم أو التخطيط وفوائد عديدة أخرى.

الكلمات المفتاحية: ضبط المخزون، نموذج كمية الطلب الثابتـة، خوارزميـة، نظـم خبيرة، ذكاء صنعي، التتبع الخلفـي، لغـة البرولـوغ، بيانـات استاتيكية، بيانات ديناميكية.

The Prolog's strength in writing the experts systems

Hend Al-Khetba , Mohammad Jamal Al –Laban and Adnan Ammuora

Department of Mathematics -Faculty of Sciences-Damascus University-Damascus

Received 25/01/2006
Accepted 02/05/2006

ABSTRACT

In this research, we try to show the major features of prolog which make it a strong expressive language about writing the expert systems and the traditional languages lacks them as Pascal language. We also provide expert system, the purpose of it is the Inventory Control by apply the Fixed-Order Quantity Model, and we clearified concept of the static and dynamic data in Prolog. Finally, we compared between the databases in prolog and some of their quires with Access and SQL.

Expert systems are considered as one of the main applications of artificial intelligence, which are known as knowledge based systems. And the expert systems are computer applications which embody some non-algorithmic expertise for solving certain types of problems. For example, the problems which provide advice, analysis, classification, diagnostic, explanation, teaching, or designing…etc.

Key Words: Inventory control, Fixed-Order Quantity Model, Algorithm, Experts systems, Artificial intelligence, Backtracking, Prolog, Static data, Dynamic data .

1- تعريفات ومفاهيم

تعريف: التسلسل التراجعي هي استراتيجية استدلال تحاول إثبات فرضية عن طريـق تجميع معلومات داعمة.

تعريف: التسلسل التقدمي هو استراتيجية استدلال تبدأ بمجموعة معروفة من الحقـائق وتشتق حقائق جديدة باستخدام قواعد تتطابق مقدماتها المنطقية مـع الحقـائق المعروفـة وتستمر هذه المعالجة حتى يتم الوصول إلى حالة الهدف، أو حتى تنتهي القواعـد التـي تملك مقدمات منطقية تتطابق مع حقائق مشتقة أو معروفة.

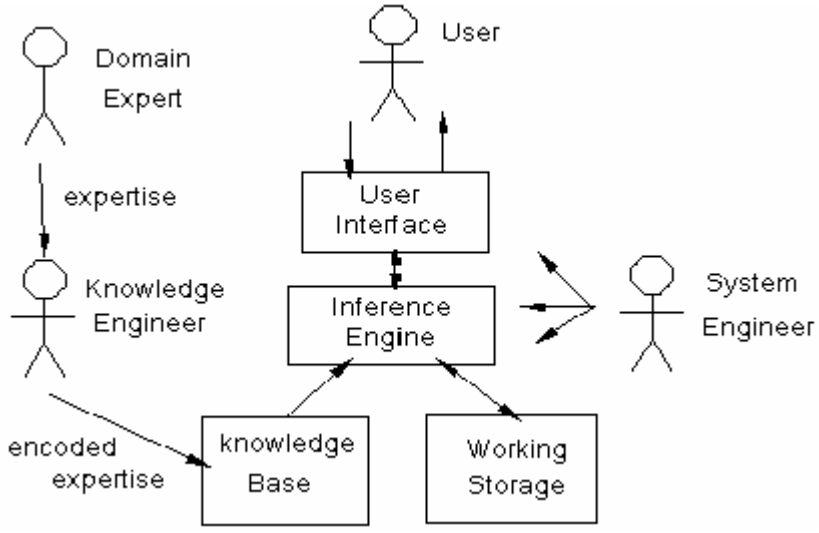
تعريف: المخزون هو المواد الموجودة في المخزن أو الموارد المستخدمة في تنظـيم ما. أما نظام المخزون فهو مجموعة سياسات وضوابط تراقب مستويات مخزون و تعـين مستويات ينبغي المحافظة عليها، ومتى يجب استكمال النقص، وكم يجب أن يكون حجـم الكمية المناسبة.

2- المكونات الأساسية للنظام الخبير

- قاعدة المعرفة- وهي تمثيل تصريحي للخبرات، تكتب غالباً بشكل قواعد IF-THEN.
- ذاكرة التخزين- وهي عبارة عن بيانات مخصصة للمسألة المراد حلها.
- محرك الاستدلال- وهو شفرة نواة النظام التي تستنتج نصائح مـن قاعـدة المعرفـة وبيانات خاصة بالمسألة في ذاكرة التخزين.
- واجهة المستخدم- وهي الشفرة التي تتحكم بالحوار بين المستخدم والنظام.

ومن الأشخاص الذين لهم دور مهم في تصميم النظام:

- خبير المجال- وهم الأشخاص الخبراء الحاليين في حل المسائل المطلوب من النظـام حلها.
- مهندس المعرفة- وهو الشخص الذي يشفر معرفة الخبير بـشكل تـصريحي يمكـن استخدامه من قبل النظام الخبير.
- المستخدم- وهو الشخص الذي سوف يتشاور مع النظام للحصول على نصيحة التي قد قدمت من قبل الخبير (الشكل 1).



الشكل (1) العلاقة بين مكونات النظام الخبير والأشخاص الذين يتعاملون معه.

3-ميزات النظام الخبير:

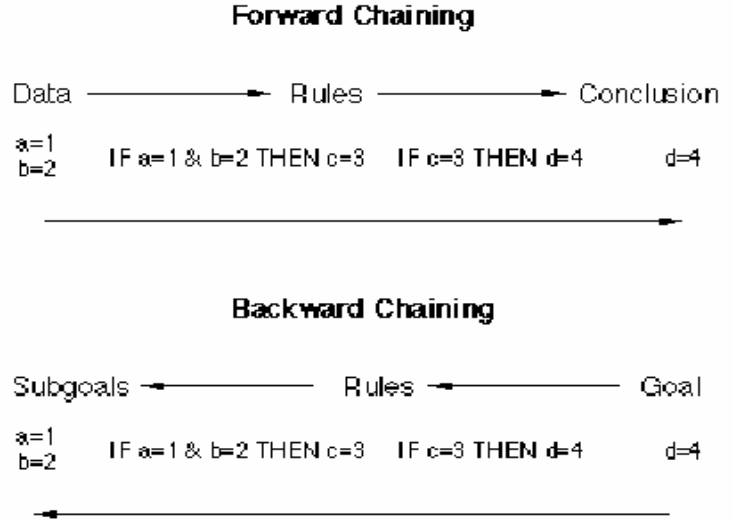
- الاستنتاج المقاد بالأهـ ـداف (goal driven reasoning) أو التسلـ ـسل الخلفي (backward chaining).

يبدأ نظام التسلسل التراجعي بهدف يريد إثباته فيفحص أولاً ذاكرة العمل ليرى إذا كان الهدف قد أضيف سابقاً. إذا لم يثبت سابقاً فإن النظام يبحث في قواعده عن قاعدة أو أكثر تحوي الهدف في الجزء Then الخاص بها. يدعى هذا النوع من القواعد بقاعدة الهدف ثم يتحقق النظام من أن المقدمات المنطقية لقاعدة الهـ ـدف موجـ ـودة فـ ـي ذاكـ ـرة العمـ ـل. المقدمات غير الموجودة تصبح فيما بعد أهدافاً جديدة (تدعى أهدافاً جزئية) يجب إثباتهـ ـا بدعم من قواعد أخرى. تستمر هذه العملية بأسلوب تعاودي حتى يجد النظام مقدمة منطقية غير مدعمة من أي قاعدة وتدعى هذه المقدمة المنطقية بالمقدمة الأصلية. وهي المقدمـ ـة النظرية لقاعدة غير مستنتجة من أي قاعدة أخرى.

وهي تقنية استدلال تستخدم قواعد IF THEN لتجزئة الهدف بـ ـشكل تكـ ـراري إلـ ـى أهداف جزئية أصغر يكون إثباتها أسهل.

- الاستنتاج المقـاد بالبيانـات (data driven reasoning) أو التسلـسل الخلفي (forward chaining): وهي تقنية استدلال تستخدم قواعـد IF THEN لاستنتاج حل المسألة من بيانات أولية.

يفحص النظام كل قاعدة حسب تسلسلها ويرى إذا كانت مقدماتها المنطقية موجودة في ذاكرة العمل. عندما يجد النظام مطابقات من أجل كل المقدمات المنطقية عنـدها يـضع نتائجها المنطقية في ذاكرة العمل. ثم يكرر النظام المعالجة خلال دورة جديدة. ويتوقـف النظام عن العمل عند عدم وجود قواعد جديدة يمكن أن تشتغل. (الشكل 2)



الشكل (2) الفرق بين التسلسل التقدمي والتسلسل الخلفي

4- لمحة عن لغة البرولوغ

تعد لغة البرولوغ لغة برمجة قوية، تعتمد على أسس منطـق الإسـناد مـن المرتبـة الأولى. ويأتي اسم اللغة من العبارة الفرنسيةPROgrammation en LOGique. صممت اللغة من قبل Colmerauer وP.Roussel في السبعينيات في جامعـة مرسـيليا. توجـد التطبيقات الأساسية للغة البرولوغ في مجال الذكاء الصنعي، كما توجـد البرولـوغ فـي مجالات أخرى تكون فيها المعالجة الرمزية من اهتماماتها الأولية. يتألف نظام البرولـوغ من مركبتين قاعدة بيانات البرولوغ ومفسر البرولوغ. برنامج البرولوغ هـو بالأسـاس برنامج منطقي يتألف من عبارات هورن التي تدخل إلى قاعدة بيانات البرولوغ من قبـل

المبرمج. ومفسر البرولوغ يؤلف طريقة الاستدلال التي تعتمد على الحـل (resolution) SLD.

البرولوغ لغة برمجية خاصة بالحسابات الرمزية غير الرقمية وضعت من أجل حـل المسائل التي تتضمن عناصر وعلاقات بين العناصر. عناصر البيانات فـي هـذه اللغـة تدعى تعابير terms. التعبير هو إما ثابت أو متغير أو تعبير مركب (حقائق وقواعد).

-الثوابت تتضمن الأعداد الصحيحة والأعداد الحقيقية وذرات atoms مثـل a، void، =، =:، 'Algol-68'، [].

-المتغيرات وهي سلسلة أحرف أبجدية تتضمن (_) تبـدأ بحـرف كبيـر مثـل X، Value،A، A1، 3_، _A.

-بنى Structures وهي ثلاثيات من تعابير تسبق بذرة وهو المعامل functor .
مثال: .vec3(0,1,0), mat2(vec2(1,0), vec2(0,1))

-القوائم Lists سلاسل وهي النوع الفعلي للبنى لكنها تملك تركيباً خاصاً.
مثال: [the,dog,bit,the,cat], [], [[1,0,0], [0,1,0], [0,0,1]]

-الأشرطة Strings: تكتب الأشرطة عادة كسلسلة من المحارف المحاطة بأقواس.

-الحقائق Facts: تعد الحقائق الشكل الأبسط لإسـنادات البرولـوغ وهـي مـشابهة للـسجلات فـي قواعـد البيانـات العلائقيـة. التركيـب اللغـوي Syntax للحقيقـة: Pred(arg1, arg2, …..,argN). حيث Pred اسم الإسنادية و arg1,…..,argN الوسـطاء وN يدعى arity وتنتهي كل عبارات البرولوغ بنقطـة. الإسـنادية ذات arity (0) هـي pred. يمكن أن تكون الوسطاء أي تعبير قانوني في البرولوغ (عدد صحيح موجـب أو سالب- ذرة- متغير- بنية). تستخدم الحقائق لتخزين البيانات التي يستخدمها البرنامج.

-القواعد Rules: تُصرح القواعد عن أشياء أو علاقات تكون صحيحة اعتماداً علـى تحقق شرط معين.

الشكل العام لها: head: body حيث head هو تعريف إسناد يشبه الحقيقة أما (:) فهو رمز العنق يقرأ "إذا" أما body فهو هدف أو أكثر (اسـتعلام). يمكـن أن يكـون body خالياً أي من الشكل: .P وإذا لم يكن body خالياً فيعبر عن القاعدة بالشكل: P:- Q, R, S. حيث P هدف الرأس وQ, R, S أهداف تكون body. يمكن أن تقرأ هذه القواعد تصريحياً بالشكل: تكون P صحيحة إذا كانت Q وR وS صحيحة أو بشكل إجرائي بالشكل: لتحقيق الهدف P يجب أن تتحقق الأهداف Q وR وS.

-إدارة البيانات managing data: يسمح البرولوغ لنا بمعالجة القاعدة بشكل مباشـر ويقدم إسناديات مبنية داخلياً لإنجاز هذه الوظائف من الإسناديات الأساسية:

asserta(X) تضيف العبارة X كعبارة أولى لإسنادها. assertz(X) تضيف العبارة X كعبارة أخيرة لإسنادها.

retract(X) تحذف العبارة X من قاعدة البيانات.

5-مظاهر قوة البرولوغ

5-1-rule-based programming: تتألف قاعدة البيانات أو قاعدة المعرفـة فـي لغة البرولوغ من مجموعة حقائق وقواعد تكتب في ملف نصي. وتعـد القواعـد اللبنـة الأساسية في كتابة برامج البرولوغ إذ تستخدم في بناء الاستعلامات التي تربط بين عـدة علاقات ممثلة بحقائق أو قواعد أخرى كما في القاعدة:

supervises(X,Y):-manager(X),clerk(Y). التي تربط بين حقيقتـين. ويمكـن أن تستخدم البرولوغ لكتابة برامج من النوع الإجرائي باستخدام قواعد خاصة تنفـذ أهـدافها الفرعية كأنها تعليمات لغة إجرائية ولكنها ليست تعليمات بل أهـداف يمكـن أن تكـون صحيحة أو مغلوطة كما في القاعدة:

go:-write('What is your name?'), read(Name),check(Name).

هذه القاعدة الإجرائية تقرأ اسماً ما وتختبر كون صاحب هذا الاسم ذكراً أو أنثى مـن خلال القاعدة check:

check(Name):-boy(Name),write('this is a boy'),nl.

check(Name):-girl(Name),write('this is a girl'),nl.

كما يمكن أن تستخدم القواعد لتعريف عناصر معينة كما في قاعدة البيانات:

rule(1,orange):-color(orang),skin(thick).rule(2,apple):-color(red),shape(round).

rule(3,bannana):-color(yellow),shape(long).rule(4,melon):-color(green),size(larg).

rule(5,date):-size(small),shape(long). rule(6,'not known').

نلاحظ من قاعدة البيانات هذه أن كل قاعدة تعرف نوعاً من الفاكهة. تتم معالجة هـذه القاعدة بلغة البرولوغ من خلال قراءة جميع الخصائص المشتركة وغير المشتركة بـين جميع الأنواع وحشرها بشكل مؤقت في ذاكرة العمل من خلال الإسنادية المبنيـة داخـلاً

assert. ثم حذفها منها بعد التعرف على نوع الفاكهة من خلال الإسنادية المبنيـة داخـلاً retract.

لو حاولنا تعريف هذه العناصر في لغات أخرى تقليدية مثل الباسكال لاحتجنا إلى بنى معطيات معقدة كالقوائم والمؤشرات والملفات والسجلات. كما في المقطع البرمجي الآتي الذي يقوم ببناء قاعدة بيانات تعرف مجموعة من الكائنات (ممكن أن تكـون فواكـه أو أشكالاً هندسية أو حيوانات) كجزء من نظام خبير بسيط:

```
Type
pint=^att;
att=record
  atr:string[15];
  next:pint;
end;
r=record
   name:string[15];
   at:pint;
end;
var
p,q:pint;attribute:string[20];
f:file of r;s:r;
k,i:integer;
begin
   assign(f,'fob.dat');
   rewrite(f);
   with s do
      begin
         write('enter name');
         readln(name);
         while name <> 'zzz' do
            begin
               write('enter att');
               new(p);p:=nil;
               readln(attribute);
                  while attribute <>'z' do
                  begin
                  new(q);q^.atr:=attribute;
                   q^.next:=p;p:=q;
                   write('enter att');
                   readln(attribute);
                   end;
               at:=p;   write(f,s);
               write('enter name');
               readln(name);
            end;
        end;
close(f);
end.
```

نلاحظ أننا استخدمنا بنية معطيات ديناميكية معقدة تتمثل بملف سجلات. يتضمن كـل سجل حقلين الأول من نوع شريطي لتعريف اسم الكائن والحقل الثاني من نوع قائمة عدد عناصرها متغير حسب خصائص كل كائن. وللتعامل مع معطيات ملف الـسجلات نقـوم بكتابة كتلة برمجية جديدة كما في المقطع البرمجي الآتي الذي يقوم بإظهـار محتويـات الملف.

```
program tt;
type
Begin
assign(f,'fob.dat');
```

```
reset(f);
while not eof(f) do
begin
read(f,s);
writeln(s.name);
while s.at <> nil do
begin
writeln(s.at^.atr);s.at:=s.at^.next;
end;end;
close(f);
readln;
end.
```

```
pint=^att;
att=record
  atr:string[15];
  next:pint;
end;
r=record
   name:string[15];
   at:pint;
end;
var
f:file of r;s:r;
k,i:integer;q:pint;
```

مما سبق نجد صعوبة تمثيل الحقائق والتعامل معها بلغة تقليديـة مقارنـة مـع لغـة البرولوغ.

2-5- تنفيذ التعقب الخلفي backtracking execution: يعد التعقب الخلفي الآلـي من أكثر الصفات المميزة للبرولوغ والتي تجعلها لغة طيعة أكثر من اللغات التقليدية، ومن ثم يمكن معالجة الاستفسارات (الأسئلة) بشكل أسهل لأن المترجم يقوم بالجزء الأكبر من العمل. فإذا كانت البرولوغ تعمل على الأهداف الفرعية لقاعدة ما وأخفق أحدها فإنهـا لا تستسلم مباشرة ولا تجعل القاعدة تخفق، فتقوم بالتعقب الخلفي إلى الهدف الفرعي السابق وتجرب أمثلة أخرى عنه في قاعدة المعطيات ثم تتحرك من جديد إلى الأمام. ومـن ثـم يمكن القول: إن البرولوغ تجرب كل الحلول الممكنة أتوماتيكياً للهدف قبل أن تخفق.

لتوضيح التعقب الخلفي في لغة البرولوغ نفرض وجود قاعدة البيانات الآتية المكونـة من الحقائق الآتية:

```
location(desk, office).location(apple, kitchen).
location(flashlight, desk). location('washing machine', cellar).
location(nani, 'washing machine'). location(broccoli, kitchen).
location(crackers, kitchen).location(computer, office).
```

وأنه لدينا الاستعلام:

```
? location(apple, kitchen).
```

تتم معالجة هذا الاستعلام بالشكل الآتي:

-يبدأ المترجم بالجملة الأولى محاولاً مطابقة نمـوذج الاسـتعلام والجملـة الأولـى الموجودة في قاعدة البيانات ولكن عملية التوحيد تخفق. ثم يتم الانتقال إلى الجملة الثانيـة التي ينجح عندها التوحيد ويعطي المترجم الإجابة X=apple .

-إذا طلب إجابات أخرى عندها يلغي توحيد المغير X و يتابع البحث باستخدام الجملة التي تلي الجملة التي حققت الاستعلام أي يبدأ من الجملة الثالثـة وهنـا يخفـق التوحيـد وتستمر المطابقة إلى أن يصل إلى الجملـة الـسادسة التـي تـنجح بإعطـاء الإجابـة X=broccoli.

أما استخدام التعقب الخلفي مع القواعد فيتم بالشكل الآتي: لتكن قاعدة البيانات الآتية:

```
clerk(jones). clerk(smith).
typist(brown).
manager(patel). manager(lee).
super(X,Y):-manager(X), clerk(Y).
super(X,Y):- clerk(Y), typist(Y).
super(X,Y):-manager(X), typist(Y).
```

ويفرض لدينا الاستفـسار الآتـي: super(Super, brown). -? للإجابـة عـن هـذا الاستفسار تقوم البرولوغ بالخطوات الآتية:

-يفحص القاعدة الأولى نجد تحقق السؤال الأول من أجل X=patel وإخفاق الـسؤال الثاني ولكن البرولوغ لا تخفق بل يبدأ التعقب الخلفي بالعودة إلى السؤال الأول والانتقـال إلى manager(lee) ثم تجرب clerk(brown) التي تحقق مرة ثانيـة ومـن ثـم تخفـق القاعدة الأولى لأن جميع أهدافها الفرعية أخفقت.

-تنتقل البرولوغ إلى القاعدة الثانية فتجد clerk(jones) ويتحقق الهدف الفرعي الأول وينتقل إلى الهدف الفرعي الثاني فيجد typist(brown) فتنجح القاعـدة الثانيـة وتعطـي الجواب Super=jones.

إن ما تقوم به البرولوغ من تعقب خلفي لا يمكن أن تقوم به لغة تقليدية لأنه لا توجـد وسيلة لقراءة عنصرين من بنية المعطيات نفسها معاً، فمثلاً لو مثلنـا قاعـدة المعطيـات السابقة بملف سجلات كل سجل يحوي حقل الاسم وحقل الوظيفة وأردنا قراءة عنـصرين من الملف بشكل متسلسل ومقارنة وظيفتيهما لما تمكنا من ذلك أبداً.

5-3-استخدام cut أو (!) وهي إسنادية مبنية داخلاً تمكن المبرمج من التحكم بتنفيـذ البرنامج وتمنع التعقب الخلفي دون حفظ المعلومات بحيث يصبح البرنامج أكثر فعالية من حيث استهلاك الوقت والذاكرة. أي أنها تقوم بعملين: الأول: تقوم بإزالة كـل العلامـات والمعطيات للأهداف التي تحققت من المكدس وتلغي احتمال انهياره وتـوفر مكانـاً فـي الذاكرة. الثاني: توفر الوقت بإيقاف التعقب الخلفي فإذا كان هناك تعقب خلفي في مرحلـة لاحقة وعاد إلى cut فإنه يمنَع من تجاوزها وهذا يجعل البرنامج يعمل بشكل أسرع.

المثال الآتي يوضح استخدام (!).ليكن البرنامج الآتي الخالي من (!):

```
s(X,Y) :- q(X,Y).
s(0,0).
 q(X,Y) :- i(X),j(Y).
i(1). i(2). j(1). j(2). j(3).
```

وليكن الاستعلام الآتي:.?- s(X,Y) النتائج تكون بالشكل:

```
X = 1  Y = 1 ;  X = 1  Y = 2 ; X = 1  Y = 3 ;  X = 2  Y = 1 ;  X = 2  Y = 2 ;
 X = 2  Y = 3 ;   X = 0   Y = 0;
no
```

بإضافة (!)إلى القاعدة الثالثة.q(X,Y) :- i(X),!,j(Y)

عندها تكون نتيجة تنفيذ الاستعلام الآتي هي:

```
?- s(X,Y).
 X = 1  Y = 1 ;  X = 1  Y = 2 ;  X = 1  Y = 3 ; X = 0Y = 0;
 no
```

نجد أن وضع (!) منع تنفيذ السؤال الأول من القاعدة أكثر من مرة أي أن cut تعني حلاً وحيداً لهذا السؤال.

5-4-مطابقة النموذج المبني داخلاً built-in pattern matching. عندما تحاول البرولوغ التحقق من أهداف إسنادية، فإنها تبحث خلال الجمل التي تعرف هذه الإسنادية وعندما تجد مطابقة من أجل متغيراتها فإن البرولوغ تضع علامة على الجملة التي استخدمت لتحقيق الهدف. وإذا طلب المستخدم أجوبة أخرى فإن البرولوغ تستأنف بحثها في الجمل في المكان المعلم. وهذا يتم في اللغات التقليدية باستخدام عمليات مقارنة طويلة وقد تكون معقدة.

6-نظام خبير "ضبط المخزون"

Inventory Systems: تقسم نظم المخزون إلى قسمين نظام مخزون وحيد الفترة Single-Period Inventory Models ونظام مخزون متعدد الفترة Multi period Inventory Systems.

يرتكز هذا التصنيف على كون القرار هو قرار شراء لمرة واحدة حيث يصمم الشراء لتغطية فترة زمنية ثابتة وأن المادة لن يعاد طلبها، أو أن القرار يتضمن مادة سيتم شراؤها بشكل دوري بحيث يحفظ المخزون في المخزن ليستخدم في الطلب. يمكن لنظام مخزون متعدد الفترة أن يكون أحد نموذجين: نموذج كمية طلب ثابتة ونموذج فترة زمنية ثابتة. وسوف نتحدث فقط عن النموذج الأول موضوع دراستنا.

6-1-نموذج كمية الطلب الثابتة Fixed-Order Quantity Models: هـ ـو نظـ ـام مخزون متعدد الفترة يحاول تعيين النقطةR التي يجب عندها تقديم طلب وحجم الطلب Q. نقطة الطلب R هي دائماً عدد محدد من الوحدات. يقدم طلب من حجمQ عنـ ـدما يـ ـصل المخزون المتاح إلى النقطة R.

حالة مخزون Inventory position تعرف بالمعادلة:

Inventory position=on-hand + on-order- backordered

يركز نظام كمية الطلب الثابتة على كمية الطلب ونقطة إعادة الطلب. وفي كـ ـل مـ ـرة يتم فيها إخراج وحدة من المخزن تقارن الكمية المتبقية في المخزون مباشرة مـ ـع نقطـ ـة إعادة الطلب. فإذا انخفضت الكمية إلى هذه النقطة، فإن طلباً مقداره Q يقدم.

الخطوة الأولى في إنشاء أي نظام مخزون هي تطوير العلاقة الوظيفية بين المتغيرات التي يهتم بها النظام ومقياس الفعالية وبالنسبة لحالتنا نهتم بالكلفة لذلك تكون المعادلة:

Total =	Annual	+	Annual	+	Annual
annual cost	purchase cost		ordering cost		holding cost

أو:

$$TC = DC + \frac{D}{Q}S + \frac{Q}{2}H$$

حيث:D هو الطلب السنوي،C كلفة كل وحدة،S كلفة تقديم الطلب، Hكلفة تخزين كل وحدة.

الخطوة الثانية هي في تطوير النموذج هي إيجاد كميـ ـة الطلـ ـبَ Q_{opt} التـ ـي تكـ ـون الكلفة عندها في الحد الأدنى. باشتقاق المعادلة السابقة بالنسبة لـ Q وعنـ ـد عـ ـدم هـ ـذا المشتق نجد:

$$\frac{dTC}{dQ} = 0 + \left(\frac{-DS}{Q^2}\right) + \frac{H}{2} = 0$$

ومنه نجد:

$$Q_{opt} = \sqrt{\frac{2DS}{H}}$$

وتكون نقطة إعادة الطلب ببساطة هي:

$$R = \overline{d}L$$

حيث $\overline{d}$ هي معدل الطلب اليومي وL هي زمن الانتظار بالأيام (يمتد بين الوقت الذي يتم فيه تقديم الطلب والوقت الذي يتم فيه إيصاله للمخزن).

6-2-فكرة النظام: يوجد في مخزن (inventory) مجموعة من المواد ويتعامل مع مجموعة من الزبائن تعرف كل مادة بسبعة وسطاء رقم المادة واسمها والطلب السنوي وكلفة الطلب وكلفة التخزين وزمن الانتظار وكلفة كل وحدة.

item(Number, Name, Annual-demand, O-c, H-c, L-t ,U-c).

ويعرف الزبون باسمه ومدينته ونوع بطاقة الائتمان التي يملكها:

customer(Name, Town, Credit-rating).

هذه المعلومات عبارة عن بيانات ساكنة (ستاتيكية) أي ثابتة لا يمكن تعديلها نتيجة التنفيذ يوجد لهذا النظام بيانات ديناميكية تتغير نتيجة التنفيذ، وضعت هذه البيانات على شكل حقائق في ملف خارجي rr.pl وبدأنا البرنامج بعبارة: :-dynamic inventory/2. لجعل الحقيقة inventory(Number, Quantity). قابلة للتعديل في أثناء التنفيذ.

6-3-ما يقدمه هذا النظام

-طلب مادة معينة من المخزن، إن أمكن ذلك، من خلال القاعدة order فإذا كانت الكمية المطلوبة لا تتجاوز الكمية الموجودة في المخزن وأن الزبون الذي طلب المادة من مدينة معينة أو يحمل بطاقة ائتمان معينة عندها يكون الطلب صحيحاً.

-في حالة أن مخزون المادة أصبح أقل من الكمية الحرجة يبين النظام أن الوقت قد حان لإدخال كمية إضافية من المادة إلى المخزن ويتم ذلك مباشرة بإضافة الكمية Q_{opt} التي تجعل الكلفة الكلية بحدها الأدنى.

-في حالة أن كمية المادة المطلوبة أكبر من المخزون يبين النظام أن الطلب غير مسموح به.

-عند تحقيق الطلب يتم تحديث محتويات المخزن من المادة المطلوبة بشكل تلقائي داخل الملف rr.pl ويخزن بشكل مباشر من خلال القاعدة save.

-يمكن للنظام أن يعطي تقريراً عن المخزن في أي لحظة.

- يمكن تحديث مخزون مادة بشكل مباشر من قبل المستثمر.

-يمكن الخروج من النظام مباشرة

-يقدم النظام قائمة بمجموعة خيارات تبقى متاحة ما دمنا نريد التعامل مع النظام إلى أن نخرج منه نهائياً.

6-4-شفرة النظام

```
:-dynamic inventory/2.
run:-cls,introduction,menue.
menue:-nl,nl,                                tab(40),write('***MENUE***'),nl,
tab(40),write('==========='),nl,
   tab(40),write('  O: order  '),nl, tab(40),write('  U: up_date_user '),nl,
   tab(40),write('  L: list the inventory '),nl, tab(40),write('  E: exit  '),nl,
   tab(40),write('===================='),nl,
   tab(40),write(' chose the operation ?'),nl,
  tab(40),write('===================='),nl,                read(Choise),nl,
choise(Choise).
choise(o):-main,menue.
choise(u):-read(I),read(Q), update_inventory_user(I,Q),menue.
choise(l):-consult('c:/windows/desktop/rr.pl'), report_inventory,menue.
choise(e):-tab(60),write('Goodby').
choise(_):-tab(40),write('Please try again!'),menue.

cls:-put(27), put('['), put('2'), put('J').

introduction:-tab(30),
 write('      ********************************************** '), nl,
tab(30) ,
 write('    a customer order inventory application  * '), nl, tab(30),
 write('   *  you can update the inventory       * '), nl, tab(30),
 write('    *   there are rules which add intelligence to the database: '), nl,
tab(30),
 write('   *       good_customer, valid_order  * '), nl,  tab(30),
 write('   *      and there are rules which are processes:  * '), nl, tab(30),
write('   *     order, report_inventory    * '), nl,  tab(30),
 write('   ********************************************** '),  nl,nl.

initi:-consult('c:/windows/desktop/rr.pl').
```

```
main :-initi, order,save('c:/windows/desktop/rr.pl').

customer(dennis, winchester, xxx). customer(dave, lexington, aaa).
customer(ron, lexington, bbb). customer(julie, winchester, aaa).
customer(jawaid, cambridge, aaa). customer(tom, newton, ccc).

item(p1,p,1000,3,1.25,4,12.5). item(p2,t,2000,4,2,4,12).
item(p3,a,1500,5,1.5,5,10). item(p4,n,1200,3,1.5,4,11).
item(p5,s,1400,5,1.2,3,11.5). item(p6,k,1300,4,1,3,10.2).
item(p7,m,1800,3,1.5,2,12).

save(FileName) :- tell(FileName),
 listing(inventory), told.

item_quant(Item, Quantity):-  item(Partno, Item,_,_,_,_,_),
 inventory(Partno, Quantity).

reorder(Item):-  item(Partno, Item,D,S,H,L,C),  inventory(Partno, Quantity),
  Q is round(sqrt(2*D*S/H)),  RP is round(D/365*L),  Quantity < RP,
  tab(50),write('Time to reorder '), write(Item),nl,nl,
  NQuantity is  Quantity+Q,  retract(inventory(Partno, Quantity)),
  asserta(inventory(Partno, NQuantity)).

reorder(Item):-  tab(50),write('Inventory level ok for... '),  write(Item), nl.
good_customer(Cust):-  customer(Cust, _, aaa).
good_customer(Cust):-  customer(Cust, winchester, _).

order:-  write('Customer: '),  read(Customer),  write('Item: '),  read(Item),
  item(Partno, Item,D,S,H,L,C),  write('the annual demand is '),write(D),nl,
  write('Quantity:                  '),                                  read(Quantity),
valid_order(Customer,Item,Quantity),nl,
  update_inventory(Item,Quantity),nl,  reorder(Item),!.

valid_order(C, I, Q):-  item(Partno, I, _,_,_,_,_),  inventory(Partno, Onhand),
  Q =< Onhand,  good_customer(C).
valid_order(C, I, Q):- item(Partno, I, _,_,_,_,_), inventory(Partno, Onhand),
write('Bad order'),nl, write(' there is  '), write(Onhand),
write('  in the inventory  '),nl, fail.
```

```
update_inventory(I,Q):- item(Pn, I, _,_,_,_,_), inventory(Pn, Amount),
  NewQ is Amount - Q, retract(inventory(Pn, Amount)),
  asserta(inventory(Pn, NewQ)), tab(50),write('the current inventory is ... '),
  write(NewQ),nl.
update_inventory_user(I,Q):- item(Pn, I, _,_,_,_,_),
consult('c:/windows/desktop/rr.pl'),
  inventory(Pn, Amount),write(Amount),nl, NewQ is Amount + Q,
  retract(inventory(Pn, Amount)), asserta(inventory(Pn, NewQ)),
  save('c:/windows/desktop/rr.pl'), tab(50),write('the current inventory is...
'),
  write(NewQ),nl.
report_inventory:- item_quant(I, Q), write(I), tab(1), write(Q), nl, fail.
report_inventory:-true.
```

ملاحظة: الملف rr.pl يحوي البيانات الديناميكية الخاصة بمخزون كـ ل مـ ادة حيـ ث يعرف المخزون بـ inventory(Number, Quantity) أي أن الملف يحـ وي البيانـ ات الآتية مثلاً:

```
inventory(p2, 18). inventory(p1, 8). inventory(p4, 23). inventory(p3, 10).
inventory(p5, 23). inventory(p6, 14). inventory(p7, 8).
```

ملاحظة: استخدمنا في برمجة النظام الخبير نسخة البرولوغ SWI-Prolog-version 4.0.2.

7- المقـ ارنـ ة

يمكن مقارنة قواعد البيانات بلغة البرولوغ مع قواعد البيانات Access.

<u>أولاً بناء قاعدة البيانات</u> على سبيل المثال لتكن لدينا قاعدة بيانات تتألف من الجـ داول الآتية:

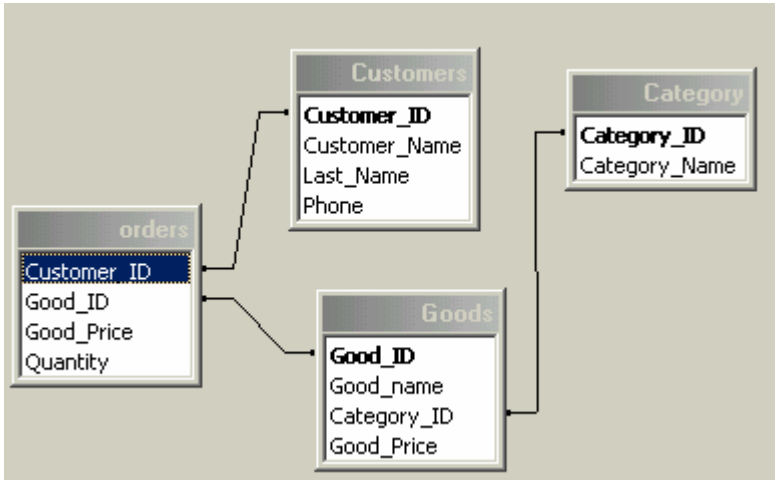
جدول Customers يحوي الحقول الآتية: ID-Cust ، Name، Last_Name، Phone

جدول Category يحوي الحقول الآتية:ID-Categ ، Name

جدول Goods يحوي الحقول الآتية: ID-Good ، Name، ID-Categ، Unit_Price

جدول Orders يحوي الحقول الآتية: ID-Cust، ID-Good، Unit_Price، quantity

تُصمم هذه الجداول باستخدام Access على عدة مراحل ومن ثم يتم إدخـال البيانـات لها. العلاقات بين هذه الجداول هي علاقة واحد لعدة وهي موضحة بالشكل:



هذه الجداول تبنى بلغة البرولوغ باستخدام حقـائق بـسيطة فمـثلاً لبنـاء الجـدول Customers نستخدم إسنادية تحمل اسم الجدول نفسه وسطاؤها هي أسماء حقول الجدول. وكذلك بقية الجداول فيكون لدينا من ثم أربع إسناديات تمثل الجداول الأربعة. ومن خلال هذه الإسناديات يتم إدخال بيانات هذه الجداول.

Customer (Cust_ID, Name, Last_Name, Phone).
category(Cate_ID,Name_Cate).
good(Good_ID, Name, Cate_ID, Unit_Price).
order(Cust_ID, Good_ID, Good_Price, Quantity).

نلاحظ أنه يوجد ارتباط بين هذه الإسناديات بسبب وجود بعض الوسطاء المـشتركة وهذا يعبر عن العلاقات الموجودة في جداول قواعد البيانات السابقة. يمكن إدخـال هـذه الحقائق في ملف خاص خارجي يحمل بواسطة التعليمة Consult لتتم قراءة محتوياتـه. على سبيل المثال قاعدة البيانات المطلوبة تحوي:

customer(1,a,k,32323). customer(2,aa,kk,54532323).
category(1,foud). category(2,clean).
good(1,rice,1,30). good(2,tay,2,130). good(3,soop,2,10). good(4,tee,1,200).
order(1,1,30,40). order(1,2,130,30). order(2,3,10,300).
order(1,1,30,40). order(1,2,130,30).

ثانياً استعلامات البحث لتبيان كيفية بناء البرولوغ للعلاقات ذاتياً وفق مبـدأ مطابقـة النموذج نحاول بناء استعلامات بلغة البرولوغ و مقارنتهـا بلغـة SQL الخاصـة بـ Access.

ليكن المطلوب الاستعلام عن طلبات الشخص ذي الرقم (1): يتم ذلك بلغة البرولـوغ بالشكل البسيط الآتي:

order(1, Good_ID, Unit_Price, Quantity).

تعتمد البرولوغ في تحقيق هذا الاستعلام على المطابقة بين اسـم الحقيقـة المطلـوب الاستعلام عنها والحقائق التي تقابلها في قاعدة البيانات ثم مطابقة الوسطاء.

أما بناء الاستعلام نفسه بلغة SQL فيمكن أن يكتب بالشكل الأكثر تعقيداً:

```
SELECT orders.Customer_ID, Goods.Good_name, Goods.Category_ID,
Goods.Good_Price
FROM Category INNER JOIN (Goods INNER JOIN (Customers INNER
JOIN orders ON
Customers.Customer_ID = orders.Customer_ID) ON
Goods.Good_ID = orders.Good_ID) ON (Category.Category_ID =
Goods.Category_ID)
AND (Category.Category_ID = Goods.Category_ID) AND
(Category.Category_ID = Goods.Category_ID)
WHERE (((orders.Customer_ID)=1));
```

استعلام آخر يطلب إظهار طلبات شخص معين من فئة معينة مظهراً رقـم الزبـون ورقم المادة واسم المادة وكميتها يكتب باستخدام قاعدة برولوغ تبحث خلال قاعدة البيانات عما يحقق المطلوب:

```
rule2(Cus,Cate):-customer(Cus,Na,_,_),category(Cate,_),
good(N,X,Cate,_),order(Cus,N,_,Q),
write(Cus),tab(10),write(N),tab(10),
write(X),tab(10),write(Q),nl,fail.
```

نلاحظ من هذه القاعدة كيفية ربط الجداول مع بعضها من خـلال وسـطاء الحقـائق المتماثلة. أما لغة SQL فتعبر عن هذا الاستعلام بالتعليمات الآتية:

```
SELECT Customers.Customer_ID, Category.Category_ID, Goods.Good_ID,
orders.Quantity
FROM Customers INNER JOIN ((Category INNER JOIN Goods ON
(Category.Category_ID = Goods.Category_ID) AND
(Category.Category_ID = Goods.Category_ID) AND
(Category.Category_ID = Goods.Category_ID)) INNER JOIN orders ON
```

```
Goods.Good_ID = orders.Good_ID) ON Customers.Customer_ID =
orders.Customer_ID
WHERE (((Customers.Customer_ID)=[enter the cust]) AND
((Category.Category_ID)=[enter the categ]));
```

ثالثاً استعلامات تعديل قاعدة البيانات يمكن حذف سجل أو حقيقة من قواعد البيانات بلغة البرولوغ باستخدام الإسنادية retract بعد تعريف الحقائق القابلة للتعديل بالإسـنادية dynamicفمثلاً لحذف سجل الزبون ذي الرقم (1) بلغة البرولوغ نكتب:

```
retract(customer(1,_,_,_)).
```

أما بلغة SQL فيتم وفق التعليمات الآتية:

```
DELETE orders.Customer_ID
FROM orders
WHERE (((orders.Customer_ID)=[enter Customers]));
```

ويمكن أن نحذف جميع سجلات جدول معاً بالإسنادية retractall.

أما التحديث فيتم باستخدام القواعد. على سبيل المثال زيادة أسعار كل المواد التي يقل سعرها عن قيمة معينة فتكون القاعدة المناسبة لذلك هي:

```
update_good(X,Y):-consult('c:/t.pl'),good(N,Na,Cate,P),P<Y,
  NewP is P+X,retract(good(N,Na,Cate,P)),
  asserta(good(N,Na,Cate,NewP)),save('c:/t.pl'),fail.

save(FileName) :- tell(FileName),
  listing(good),listing(order),listing(customer),
listing(category),told.
```

أما بلغة SQL فيعبر عنها بالتعليمات الآتية:

```
UPDATE Goods SET Goods.Good_Price = Goods!Good_Price+[enter
value]
WHERE (((Goods.Good_Price)<[enter point]));
```

وهـذا طُبـق أيـضاً فـي النظـام الخبيـر الـسابق مـن خـلال القاعـدتين update_inventory(I,Q):- ، update_inventory_user(I,Q):- حيـث يـتم تعـديل مخزون المادة في الملف rr.pl بعد أخذ طلبية معينة منها أو زيادة مخزونها في أي وقت.

مع أنه من الصعب أو ربما من المستحيل تعريف مفتاح أساسي فـي قاعـدة بيانـات البرولوغ فإنه يمكن إضافة سجل طلبية إلى قاعدة البيانات دون تكرار رقمـه ودون فـتح هذه القاعدة باستخدام القاعدتين الآتيتين، القاعدة الأولى تمنع إضافة طلب له الرقم نفـسه أي الوسيط الأول نفسه والقاعدة الثانية تضيف سجلاً وتحفظه مباشرة ضمن ملف قاعـدة البيانات T.pl

```
addord(C,N,P,M):-consult('c:/t.pl'),call(order(C,_,_,_)),
write('no').

addord(C,N,P,M):- assert(order(C,N,P,M)),
save('c:/t.pl'),!.
```

وبقاعدتين مشابهتين يمكن إضافة سجل مادة دون تكرار.

```
addgood(C,N,P,M):-consult('c:/t.pl'),call(good(C,_,_,_)),
write('no').

addgood(C,N,P,M):- assert(good(C,N,P,M)),
save('c:/t.pl'),!.
```

يمكن القول: إن إمكانية "مطابقة النموذج" الموجودة في البرولوغ تجعلها لغـ ـة مفيـ ـدة لعمليات استعادة قاعدة بيانات. وكذلك التمثيل المنتظم للبيانات والبرامج هو أيـ ـضاً ميـ ـزة مفيدة. الحجم الضخم لقواعد البيانات هو مشكلة للبرولوغ.

8- الخـ ـاتمـ ـة

نستنتج مما قدمنا أن لغة البرولوغ لغة طيعة في كتابة النظم الخبيرة مقارنة مع أي لغة تقليدية مثل الباسكال أو غيرها إذ يتطلب ذلك استخدام بنى معطيات معقـ ـدة كالمؤشـ ـرات والقوائم والسجلات وعملياتها المعقدة. وكما تبين لنا من خلال النظام الخبيـ ـر المـ ـدروس والمقارنة التي أجريناها سهولة التعبير عن قواعد البيانات باستخدام حقائق بسيطة والتـ ـي يمكن أن تكتب مباشرة داخل الشفرة وتعد بيانات ساكنة أو تكتـ ـب فـ ـي ملـ ـف خـ ـارجي وتكون بيانات ديناميكية تُحمل بأسلوب سهل باستخدام الإسنادية consult. كما يمكن تعديل هذه القاعدة من حذف أو إضافة سجل أو تحديثه من خلال الإسـ ـناديتين assert وretract مباشرة بعد تعريف بنية البيانات المراد تغييرها كبنيـ ـة ديناميكيـ ـة باسـ ـتخدام الإسـ ـنادية dynamic متبوعة بعدد وسطاء تلك البنية أو باستخدام القواعد التي يمكن أن تربط بـ ـين عدة سجلات مختلفة. والذي يمكن أن يتطلب إجراءات أكثر تعقيداً بلغات قواعد البيانـ ـات مثل SQL كما بينا في المقارنة أعلاه.

المـ ـراجع REFERENCES

Expert Systems Design & Development, 1994, by Macmillan Publishing Company.

Learn Prolog Now! - Patrick Blackburn, Johan Bos and Kristina Striegnitz, Version 1.2.5 (20030212)

Operating Management For Competitive Advantage. Mc Graw Hill الطبعة العاشرة 2004، دار النشر

htt://www.amzi.com/expert systemsinprolog/xsipfrtop.htm, Published by: Amzi! inc. OH 45036 U.S.A.

Last Updated: August 2000, web http://www.amzi.com/, 2005. Adventure in Prolog http://www.amzi.com. 2005.