知識工学と人工知能 50－6
（１９８７．１．１７）

フレーム論理

中野良平　　木山稔
（ＮＴＴ　電気通信研究所）

情報処理分野に知識工学の適用が進むに従い、知識情報管理機構の共通化の必要性が強まり、知識ベースとデータベースの一体化の研究が重要になる。

本稿は、フレームモデルと関係モデル（非正規形関係論理）を共存・融合するフレーム論理を提案する。フレームモデルの構造表現は柔軟強力であるが、検索機能があまりにも基本的である。一方、関係モデルの構造表現は単純であるが、検索力は実に強力で、アーキテクチャ上の工夫で著しい性能向上が可能である。フレーム論理はフレームモデルの知識表現力と関係モデルのデータ検索力を融合したもので、フレーム検索の高水準化と大幅な性能向上、データベースのフレームビュー等が可能になる。融合に当たり、モデル写像の考え方を導入した。また、強化した検索機能の記述力は強力である。フレーム論理の強力な検索機能は知識情報処理への応用が期待でき、関係モデルから継承したＳＩＭＤ型並列処理性は知識の規模増大に伴う性能劣化を解消する専用アーキテクチャ（ＫＢ／ＤＢマシン）への道を開く。

FRAME CALCULUS

Ryohei NAKANO　　Minoru KIYAMA

NTT Electrical Communications Laboratories
1-2356, Take, Yokosuka-shi, Kanagawa-ken, 238-03 Japan

Application of knowledge engineering techniques to information processing fields calls for integration of knowledge-bases and databases. This paper proposes a new approach named frame calculus, which integrates frame and relational model. Frames make us possible to flexibly represent knowledge ,but their access interface is so primitive; on the other hand, the structure of relational model is simple, but its operations are high-lebel and strong. Frame calculus integrates flexible knowledge representation of frames and high-level operations of relational model. An idea of model mapping is introduced for integration. Expressive power of operations,showed by several examples, is expected to provide knowledge engineers with strong means to access knowledge. Frame calculus also gives us plentiful scope for performance improvement, that is KB/DB machines.

1．はじめに

情報処理分野に知識工学の適用が進むに従い、知識情報管理機構の共通化の必要性が強まる。即ち、知識ベースとデータベースの一体化の研究が重要になるが、両者を取り巻く条件に大きなギャップがあり、簡単にはいかない。しかし、概念レベルでは双方に共通点が多々あることは早くから指摘されており[1]、モデル共通化の下地は存在する。また、知識表現法を、フレームモデルや意味ネットワーク等のグラフ表現と、述語論理等による論理表現に大別する[2]とき、一方には他方にない長所があるのだから、唯一つの表現法に拘らず、互いの良い面を生かす両者共存のアプローチ[3,4]は興味深く重要である。

本稿は、一口にいうと、グラフ表現としてフレームモデルを選び、それに論理表現として関係論理[5]の拡張版を付加・融合し、フレームモデルと関係モデルを共存させるアプローチを述べたもので、知識ベースとデータベースの一体化に向けた１つの試みである。

フレームモデルは、近年研究実用化が進むエキスパートシステム構築支援システム[6,7]の知識表現モデルとして採用され、柔軟強力な構造表現が可能であるが、一般にフレーム検索機能が単一フレームを対象とした素朴なものしかないため、その上に推論系（プロダクションシステム等）を作るときには面倒でかつ大幅な性能向上が難しく、性能上の問題の１つになっている。一方、関係モデルは、述語論理に数学的基礎を置き、モデル構造は極めて単純なものしか表現できないが、検索力は実に多彩強力であるばかりでなく、基本的には集合的操作であるのでアーキテクチャ上の工夫等で著しい性能向上の可能性がある。

本稿で提案するフレーム論理(frame calculus)はフレームモデルの知識表現力と、論理型関係データ検索言語の原形である関係論理のデータ検索力を合わせ持つもので、フレームモデルで管理する知識の利用を柔軟かつ容易にすること、フレームアクセスの大幅な性能向上を可能にすること、データベースとして管理している情報をフレームとして見せること等を狙ったものである。以下、２章でフレーム論理の骨格、特徴を、３章では両モデル共存の基盤となるフレーム構造の写像について述べる。また、４章でフレーム論理の検索法を、５章でより具体的な活用法を述べる。

2．フレーム論理の概要

2．1　モデルの骨格

フレーム論理はフレームモデルに関係モデル（関係論理）の強力な検索機能を付加・融合したものである。

まず、付加を可能にするには、関係モデルが対象とするデータ構造が存在する必要がある。換言すれば、モデルユーザがフレーム構造からイメージできねばならない。フレームモデルではスロット値が単一値に限らずリストでもよいので、関係モデル側は正規形を受け皿にするのでは十分でない。しかし、非正規形まで拡張すれば、フレーム構造に、暗黙推論及び継承推論を働かせることにより、フレーム構造から非正規形への写像が可能となる（図１）。モデルユーザがフレーム構造から非正規形構造をイメージするのは、後述の例を見ても不自然ではない。関係論理の検索はこの非正規形構造を対象に行う。一方、フレームモデルの付加手続きはフレーム構造を対象にしたものである。従って、フレーム論理においては、２種の構造と、各々に対応する操作系が存在することになる。

融合を可能にするには、２種のモデル（構造と操作系）が矛盾なく動作する必要がある。今、フレームモデルはオブジェクト指向で実現されているとする。フレームがオブジェクトとして管理され、付加手続きはオブジェクトのメソッドとして登録されるという具合である。一方、非正規形関係モデルでは、フレームを束にして集合

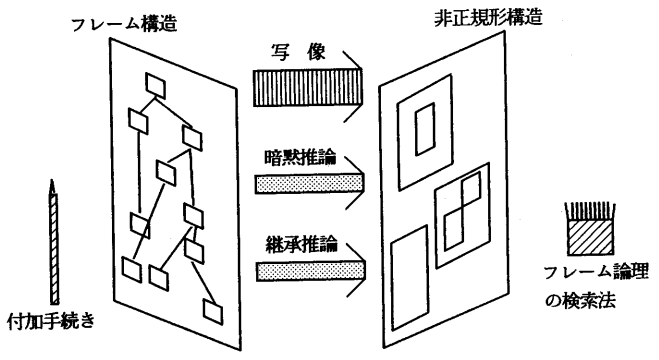

図1　フレーム論理の骨格

的演算を施すので、一見両者は反りが悪いように見える。しかし、$if_added、$if_removedの例に見るように、付加手続きは検索以外のケースで起動されることがほとんどと考えられる。動作ケースが排他的でかつ知識の共用がないとすれば、両者は常に一方しか動作しないので、動作の干渉はないと考えて良い。

また、２種の構造を共存するモデルでは、一方に対する変更が他方に正しく反映される必要がある。ここでは、変更はフレームモデルを対象に行われるから、それが非正規形構造に正しく反映されれば良い。

２種の構造の実現法としては、両方とも維持管理する方法と片方のみ維持管理し他方はビューとして実現する方法があるが、前者は維持管理が二重手間となり採りづらい。フレームモデルベースに管理し非正規形構造は必要に応じて動的に生成する方法は管理が容易であるが、生成の負荷が問題となりそうである。非正規形ベースに管理すれば、フレームモデルとしての維持管理は複雑になるが検索が効率的に行える。

２．２　モデルの特徴

フレーム論理のモデルとしての特徴を以下に要約する。

（１）フレームモデルと関係モデルの共存を可能とした。

（２）フレームモデルの長所“柔軟な知識表現力”と関係モデルの長所“強力な検索力”を合わせ持つ。従って、検索に関しては関係モデルの特長ＳＩＭＤ型並列処理性を有す。即ち、データの集合的操作が可能である。

（３）非正規形関係モデルを包含する。関係モデルの従来の常識であった正規形だけでなく、近年エンジニアリング分野のトピックである非正規形(Non First Normal Form)をも包含する。

３．フレーム構造の写像

３．１　フレーム構造の基本概念

フレーム論理では以下に述べるようなフレーム構造を対象とする。なお、付加手続きには言及しない。

（１）フレーム

モデル化対象の基本単位をフレームと呼ぶ。フレームはクラスとインスタンスに大別できる。

（２）クラスとインスタンス

同種のフレームの共通特徴に着目して一般化したフレームをクラスと呼ぶ。また、クラスとして集約された元の個々のフレームをインスタンスと呼ぶ。インスタンスとクラスの直属関係をinstance_of関係と呼ぶ。

【所属制約１】　どのインスタンスもいずれか１つのクラスに直属する。

（３）スロット

フレームの特徴の管理単位をスロットと呼ぶ。スロットの構造は次のいずれかとする。

・単一値　　・単一値の繰り返し

・副スロットの組　　・副スロットの組の繰り返し

副スロットはスロットを更にブレークダウンした単位で、このブレークダウンをスロットの階層化と呼ぶ。

（４）階層関係（is_a関係）

クラス間に上位／下位の階層関係が設定できる。上位クラスと下位クラスは、概念の上位と下位に対応する。クラス毎に複数の上位クラスを取り得る。

【所属制約２】　インスタンスは直属のクラスだけでなく、そのクラスの上位（上位の上位等を含む）にあるクラスにも属す。

（５）継承

上位／下位の階層関係は次のような性質の継承を伴う。

【継承制約１】　上位クラスのスロットはその下位（下位の下位等も含む）クラスのスロットでもある。なお、下位クラスにおいて、継承するスロットに副スロットを追加することができる。

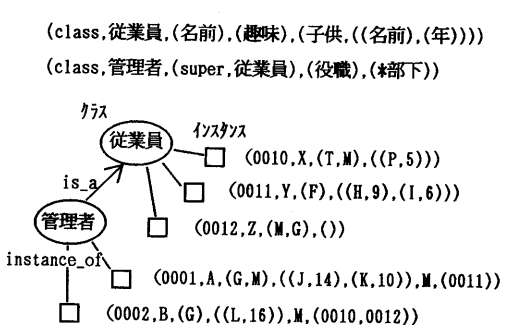

図2　フレーム構造の例

【継承制約２】　インスタンスは直属のクラスのスロットを引き継ぐものとし、インスタンス独自のスロットは取れない。
【継承制約３】　インスタンスのスロット値は、同インスタンスに規定があればその値とするが、そうでないときは、直属のクラスのスロット値とする（暗黙推論）。そこにも規定がない場合には、そのクラスの上位クラスから継承する（継承推論）。多重継承は深さ優先とする。

（６）関連(relationship)

インスタンス間の関係を関連として表現できる。関連は、参照する主体であるインスタンスのスロット／副スロット値に、参照されるインスタンスのidを代入する形で表現する。関連を表わすスロットはアスタリスク（＊）を付して規定する。
【id制約】　インスタンスidは変更しない。
【参照制約】　参照されるインスタンスは必ず存在しなければならない。

３．２　非正規形構造への写像

前節で規定したフレーム構造は以下に述べるように非正規形構造に写像できる。
［定理］　任意のフレーム構造Ｓは、クラス、インスタンスを各々関係スキーム、組として継承を施した非正規形構造Ｓ’と次の意味で等価である。即ち、任意の検索Ｑと、任意の解釈Ｉに対し、以下が成立する。

Ｑ［Ｓ（Ｉ）］＝Ｑ［Ｓ’（Ｉ）］

ただし、Ｑ［Ｘ］はＸに対する検索Ｑの結果を表わす。

（証明）

Ｓ中の任意のクラスF_cについて、以下が成立する（図３）。

- F_cのスロットは、自身に規定されたスロットと、上位クラスから継承したスロットから成る。（【継承制約１】より）
- F_cに直属するインスタンスはF_cのスロットを継承する。（【継承制約２】より）
- F_cに直属するインスタンスは暗黙推論、継承推論により適宜スロット値を継承する。（【継承制約３】より）
- F_cの下位クラスはF_cのスロットを有す。（【継承制約１】より）
- F_cの下位クラスに直属するインスタンスはF_cのスロットを有す。（【継承制約２】より）
- F_cの下位クラスに直属するインスタンスは暗黙推論、継承推論により適宜スロット値を継承する。（【継承制約３】より）
- F_cの下位クラスに直属するインスタンスはF_cに属す。（【所属制約２】より）

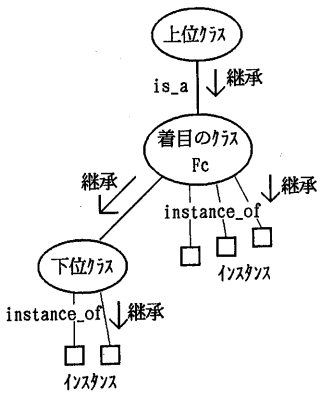

図3　クラスとインスタンスの関係

以上より、F_cに対して、F_cのスロット＋インスタンスidを属性とするような関係スキームを考え、F_cに属すインスタンスをその組とすることにより、一意に非正規形関係が作れる。一方、【所属制約１】より、どのクラスにも属さないインスタンスはないので、Ｓ（Ｉ）が有すインスタンスに関する情報はＳ’（Ｉ）のものと同等である。

Ｑ［Ｘ］はＸのインスタンスの集合によって決まるので、

Ｑ［Ｓ（Ｉ）］＝Ｑ［Ｓ’（Ｉ）］　　（証明終）

さて、上記定理に基づくＳ（Ｉ）からＳ’（Ｉ）への写像は、実は１：１ではなくｎ：１である点に留意。それは、継承のためである。暗黙推論、継承推論は逆に辿れない。従って、逆写像は一意でないが、単に、関係の各属性に対し同一名のスロットを対応させることにより、以下の性質を持つフレーム構造に写像することはできる。

- 生成されたクラス間には階層関係が全くない。
- クラススロットは暗黙値を持たない。

このことから、フレーム構造は関係のネストレベルが高々１の非正規形構造を包含することが分かる。

従業員
(01)

Iid	名前	趣味	子供 名前	子供 年
0010	X	T M	P	5
0011	Y	F	H I	9 6
0012	Z	M G	-	-
0001	A	G M	J K	14 10
0002	B	G	L	16

管理者
(02)

Iid	名前	趣味	子供 名前	子供 年	役職	部下
0001	A	G M	J K	14 10	M	01_0011
0002	B	G	L	16	M	01_0010 01_0012

図4　写像された非正規形構造の例

図４に、図２のフレーム構造を写像した結果を示す。クラス管理者がクラス従業員の下位クラスであるので、関係管理者は、関係従業員の属性を含む。それだけでなく、インスタンスid＝0001,0002については、属性値も重複している。しかし、この写像結果はモデルユーザが描くイメージであり、実現する場合には、継承して得た情報はポインタの形で管理することになろう。

４．フレーム論理の検索法

４．１　基本要素

フレーム論理の検索法は、関係論理を非正規形[8]に拡張したものである。以下に基本要素を説明する。ただし、関係モデルの用語を使用する。

（１）アルファ

派生関係を表わすアルファは目標リスト、範囲規定、条件式から成る。

目標リスト ： 範囲規定 ： 条件式

これは、ＳＱＬのSelect-From-Where構文に対応する。関係スキームもアルファである。フレーム論理では、関係のネストおよび関連(relationship)に対応して、組変数［属性名］もアルファとなり得る点に留意。

（２）目標リスト

目標リストの要素は、正規形のときは値表現のみであったが、非正規形ではアルファも許容する。これは、非正規形への拡張における重要なポイントであり、この拡張によりアルファの多段のネストが可能となる。なお、目標リスト中のアルファが空になる場合は目標リストから除外する点に注意。なお、属性を属性名で識別するので、アルファがネストする場合には、目標リストの要素（属性）の命名機能も必要になる。

（３）範囲規定

範囲規定は１個以上の範囲式の集まりである。範囲式は次の形式である。

範囲アルファ（組変数）

範囲アルファは次のように再帰的に定義する。アルファの内容は異なるが定義は関係論理と同じである。

ａ）アルファは範囲アルファである。

ｂ）Ｒ１，Ｒ２を共に範囲アルファとするとき、以下も範囲アルファである。ただし、Ｒ１とＲ２は論理和等が可能な構造でなければならない。

・Ｒ１∧Ｒ２　　・Ｒ１∨Ｒ２　　・Ｒ１∧～Ｒ２

ｃ）上記以外は範囲アルファでない。

（４）条件式

条件式の定義（以下）は関係論理と同じである。

ａ）単位式<値表現> <比較演算子> <値表現>は条件式である。

ｂ）空式は値が真の条件式である。

ｃ）Ｆが条件式のとき、～Ｆも条件式である。

d）F1,F2が共に条件式のとき、F1∧F2,F1∨F2は共に条件式である。

e）R,Fが各々範囲式,条件式のとき、（∃R）F,（∀R）Fは共に条件式である。

f）上記以外は条件式でない。

フレーム論理に特徴的な点は、関連を利用した暗黙の組参照にある。今、組変数［属性値1］が関連によって生成された組参照のとき、組変数［属性値1］［属性値2］は、参照された組の属性2を表わすといった具合である。

（5）値表現

値表現は、定数、属性値、統計関数、または、それらを四則演算で結合したものである。フレーム論理では、関係のネストに対応して、副属性値も値表現である。なお、アルファが1属性から成るとき、組変数［属性名］の代わりに、単に組変数を書いても良い。

（6）集約関数(aggregate function)

関係論理の集約関数は以下の形式で、アルファの形を保持して、指定属性に関し集約関数を作用させる。

関数名［属性名］（アルファ）

フレーム論理の集約関数も同じ。なお、関数名には、count,min,max,sum,avgがある。

4．2　記述例

図4（即ち、図2）の構造を用いて、関係論理の検索法の記述例を幾つか示す。その記述力の強力さは非正規形モデルの検索が如何に柔軟に行えるかということに他ならない。また、論理による知識表現で問題になる計算（評価）可能性については、前記の定義に基づくアルファは評価可能（Ullman[9]流でいうと安全）であることが示せる。

＜例1＞ゴルフが趣味の従業員は誰か。

(u[名前]):従業員(u):∃(u[趣味])(w)(w=G)

解説…趣味は複数あるので、その中にゴルフがある者を選ぶ。組変数wは1属性なのでそのまま書く。従業員が範囲なので、下位クラス管理者のインスタンスも対象になる。また、u[趣味]はアルファであり、存在限量子の範囲となる。

＜例2＞子供のいない従業員のインスタンスidを求めよ。

(u[Iid]):従業員(u):count(u[子供])=0

解説…インスタンスidも属性なので検索対象にすることができる。集約関数countの引数u[子供]はアルファである。

＜例3＞管理者の名前と部下の数を求めよ。

(u[名前],count(u[部下])):管理者(u):()

解説…集約関数を目標リストに書くこともできる。集約関数countの引数u[部下]はアルファである。

＜例4＞従業員の子供の平均年令を求めよ。

avg[年]((u[名前],w[名前],w[年]):従業員(u),u[子供](w):())

解説…従業員の子供の年令を取り出し、集約関数avgの引数となるアルファを作る。その際,目標リストには、u[名前],w[名前]も取り出し、重複の発生を防ぐ。

＜例5＞趣味が一致する管理者と部下及び一致する趣味リストを求めよ。

(u[名前],v[名前],(w:(u[趣味])(w),(v[趣味])(x):w=x)):管理者(u),(u[部下])(v):()

解説…組変数vが対応するアルファは関連により参照される従業員の組から成るアルファである。目標リスト中のアルファが空のとき、即ち一致する趣味がない管理者と部下の組合わせは捨てられる。

＜例6＞従業員を対象に、趣味と愛好者のリストを作れ。

(w,((x[名前]):従業員(x):∃(x[趣味])(y)(y=w))):(v:従業員(u),(u[趣味])(v):())(w):()

解説…出力用にこんなこともできる。組変数wが対応するアルファは趣味の一覧に他ならない。目標リストにアルファを書くことにより、趣味毎の愛好者リストができる。

5．フレーム論理の活用法

フレーム論理の役割として以下が考えられる。

（a）規模の大きい知識／データベースの一元格納管理機構

フレーム論理では、データベースへの写像を可能にするために、ある程度フレーム構造を限定している。このため制約されると困るような使い方には都合が悪いが、一方では関係モデルの集合的操作が可能となるため、規模の増大に対処することが容易になった。規模の大きい知識／データベースの一元格納管理機構として期待できる。

（b）推論系からの集合的な知識アクセスの効率化

プロダクションシステムにおける条件部(LHS)やprologにおけるユニフィケーション等では、データの集合的操作が可能である。適切なアーキテクチャにサポートされたフレーム論理システムはこれらを効率的に実現できる可能性がある。

（c）既存データベースのフレームビューの実現

既存の正規形データベースに対し非正規形ビューを形成して、フレーム構造と同等の扱いで検索できるようにすることも可能である（図5）。その際、フレーム論理の検索をデータベースの検索に変換する機構が必要になる。その変換は非正規形ビューを利用して行うことができる。

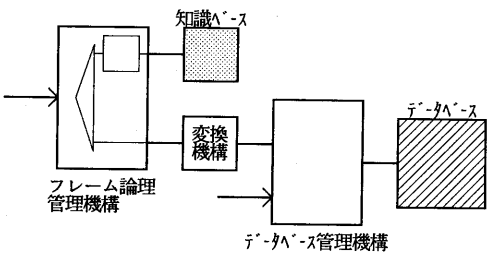

図5　データベースの知識ビュー化

6．おわりに

知識ベースとデータベースのモデル一体化の試みとして提案したフレーム論理は、フレームモデルと非正規形に拡張した関係論理を融合したモデルである。融合に当たっては、モデル写像の考え方を導入しフレーム構造を非正規形構造にマッピングした。フレーム論理は、フレームモデルであるとともに、関係ネストレベルが高々1の非正規形関係モデルを包含する。フレーム論理で強化した検索記述力は所期の通り強力であることが確認できた。フレーム論理はその柔軟な知識表現力と強力な検索機能により、エキスパートシステムを始めとした知識処理への応用が期待できる。また、関係モデルから継承したＳＩＭＤ型並列処理性は、知識（情報）の規模増大に伴う性能劣化を解消する専用アーキテクチャ（ＤＢ／ＫＢマシン）への道を開く。

参考文献

[1] Wong,H.K.T., Mylopoulos,J.:Two Views of Data Semantics:A Survey of Data Models in Artificial Intelligence and Database Management, Infor. 15,3 (1977).

[2] 松本裕治:知識表現-論理的アプローチに焦点を当てて-,情報処理, Vol.27, No.8 (1986).

[3] Rich,C.:Knowledge Representation Languages and Predicate Callculus:How to have Your Cake and Eat It Too, AAAI-82, Penn. (1982).

[4] Brachman,R.J. ,et al.:Krypton :A Functional Approach to Knowledge Representation, Computer, Vol.16, No.10 (1983).

[5] Codd,E.F.: Relational Completeness of Data Base Sublanguages, in Data Base Systems, Courant Computer Symposium 6, Prentice Hall (1972).

[6] 服部文夫、他:知識ベース管理システム(KBMS),情処研究会資料85-AI-41 (1985).

[7] Ogawa,Y ,et al.:Knowledge Representation and Inference Environment:KRINE, … an Approach to Integration of Frame , Prolog and Graphics, Proc. ICFGCS-84 (1984).

[8] Jaeschke,G., Schek,H.-J.: Remarks on the Algebra of Non First Normal Form Relations,Proc. of ACM Sympo. on Principles of Database Systems, Calif. (1982).

[9] Ullman,J.D. : Principles of Database Systems, Second Edition, Computer Science Press (1982).

付録　フレーム論理のシンタクス（フレーム構造と検索のみ）

ＸＸＸsは１個以上のＸＸＸをカンマで区切ってつないだものを表わす。

（１）フレーム構造

```
<frame> ::= <class> | <instance>
<class> ::= (c l a s s,<class name>,<class slots>)
  <class slot> ::= (<slot name>) | (<slot name>,<constants>)
                 | (＊<slot name>) | (＊<slot name>,<instance ids>)
                 | (<slot name>,<subslot1 lists>)
                 | (s u p e r,<class names>)
  <subslot1 list> ::= (<subslot1s>)
  <subslot1> ::= (<subslot name>) | (<subslot name>,<constant>)
               | (＊<subslot name>) | (＊<subslot name>,<instance id>)
<instance> ::= (<class name>,<instance id>,<instance slots>)
  <instance slot> ::= (<slot name>,<constants>) | (<slot name>,<instance ids>)
                    | (<slot name>,<subslot2 lists>)
  <subslot2 list> ::= (<subslot2s>)
  <subslot2> ::= (<subslot name>,<constant>) | (<subslot name>,<instance id>)
  <constant> ::= <number> | <string>
```

（２）検索　　関係論理からの拡張点を網掛けで示す。

```
<alpha> ::= <class name>
          |<target list>:<range formulas>:<qualifier>
          |<tuple variable><attributes>
<target list> ::= (<target list elements>)
  <target list element> ::= <target list primary>
                          |<target list primary>－>><attribute name>
  <target list primary> ::= <value expression> | <tuple variable> | (<alpha>)
<value expression> ::= <term> | <value expression>＋<term> | <value expression>－<term>
  <term> ::= <primary> | <term>＊<primary> | <term>／<primary>
  <primary> ::= <constant>
              | <tuple variable>
              | <tuple variable><attributes>
              | <aggregate function>
              | (<value expression>)
<attributes> ::= [<attribute name>] | <attributes> [<attribute name>]
<aggregate function> ::= c o u n t (<alpha>) | <function name> [<attribute name>] (<alpha>)
  <function name> ::= m i n | m a x | s u m | a v g
<range formula> ::= <range alpha> (<tuple variable>)
  <range alpha> ::= <range term> | <range alpha>∨<range term>
  <range term> ::= <range factor> | <range term>∧<range factor>
  <range factor> ::= <range primary> | <range factor>∧～<range primary>
  <range primary> ::= <alpha> | (<range alpha>)
<qualifier> ::= <boolean term> | <qualifier>∨<boolean term> | ()
  <boolean term> ::= <boolean factor> | <boolean term>∧<boolean factor>
  <boolean factor> ::= <boolean primary> | ～<boolean primary>
  <boolean primary> ::= <atomic formula> | <quantifier><range formula><qualifier> | (<qualifier>)
  <atomic formula> ::= <value expression><comp op><value expression>
  <comp op> ::= ＝ | ＜ | ＜＝ | ＞ | ＞＝ | ＜＞
  <quantifier> ::= ∃ | ∀
```