Markov Logic Networks を用いたデータ統合メディエータの提案

中山 陽太郎† 横石 潔和† 渡辺 信幸‡

†日本ユニシス株式会社 総合技術研究所 〒135-8560 東京都江東区豊洲 1-1-1
‡株式会社リライフ・ジャパン 〒135-0016 東京都江東区東陽 5-30-13
E-mail: †‡{Yotaro.Nakayama, Kiyokazu.Yokoishi, Nobuyuki.Watanabe2}@unisys.co.jp

あらまし データ統合では，ソースデータベースを統合することにより，統合データベースにおいて結果的に整合性制約への違反が生じる可能性がある．統合データベースの整合性を保つため，違反するデータの削除や更新を行うことで修復を行う必要があるが，そのためソースデータベースの情報が失われるという問題がある．本稿では，データの矛盾した状態を許容したデータ統合を目的とし，マルコフ論理を用いたデータ統合メディエータシステムを提案する．マルコフ論理をデータ統合システムに応用することで，不整合なデータを含む統合データベースの構築を可能とし，整合性に違反するデータに対する問合せを可能とする．

キーワード データ統合，整合性制約，確率推定

1. はじめに

データ統合では，異種データベースの統合により，整合性制約違反が生じる可能性がある．統合されたデータベースは，グローバル(広域)スキーマのもとで，整合性制約が機能しなければならならず，違反を起こすデータに対して修復が必要である．しかしながら，修復のためデータの削除，更新を行うことで，ソースデータベースの情報の一部が喪失する問題がある．例えば，部門ごとに水平分散され独立しているデータベースを統合する状況において，それぞれのソースデータベースではキー制約が成立しているが，データ統合により制約を満たさないタプルが生じ，関数従属性の違反を引き起こす．整合性制約を維持するためには，グローバルスキーマの元でキーを変更するか，不要なタプルを削除することで，整合性を維持するためにデータベースを修復する必要があるが，一方でソースデータベースの情報が失われることになる．

本研究では，データ統合において，マルコフ論理ネットワーク (Markov Logic Networks, MLN) を用いたデータ統合メディエータシステムを提案し，整合性制約として関数従属性への違反を許容するデータ統合の実現性の検討を目的とする．MLN を適用してデータ統合を行うことにより，データ統合における整合性制約の違反を確率的に扱うことで，制約の違反を許容したデータ統合システムにおける問合せを可能とする．また整合性制約に違反するデータを評価するための規則を検討し，データの有効性を確率的に表現するための方式を検討する．

本稿の構成は次のとおりである．2 節で，データ統合のモデルと整合性制約について述べる．3 節では，データ統合への MLN の適用について説明する．4 節では，データ統合における整合性制約への MLN 適用における実験について説明する．5 節で実験の考察と課題を検討し，6 節は，まとめと今後の方針について述べる．

2. データ統合と整合性制約

データ統合システムでは，複数の異種データソースに分散する情報を統一的な手法で問い合わせることを目的とする．データ統合として，メディエータによるデータ統合の仕組みを仮定する．メディエータシステムでは，単一の統合されたグローバルスキーマを提供するものとする．

2.1. データ統合モデル

ここでは，データ統合のモデルとベースとなる演繹データベースの枠組みについて述べる．データ統合とは，統合元となるソースデータベースを統合し，統合データベースを構成することである．統合データベースシステムは，グローバルスキーマに対する問合せをソースデータベースの問合せに変換し，利用者にソースデータベースを意識させることなく，あたかも一つの論理的な統合データベースとして結果を返却する．データ統合をモデル化するためのフレームワークとして，メディエータ (Mediator) によるデータ統合システムを想定する．ソースデータベースとグローバルデータベースとのマッピングの形式化により，データ統合のモデルとして GAV (Global-as-View) と LAV (Local-as-View) がある (文献[1],[2]参照)．各マッピングの方式として，GAV では，グローバルスキーマをソーススキーマ上の仮想表とするのに対して，LAV では，ローカルスキーマの表をグローバルスキーマ上の仮想表とする．図 1 に GAV と LAV によるメディエータの概念図を示す．GAV は連邦型データベースシステムのような仮想データ統合のモデルとして，また LAV は DWH のような物理データ統合のモデルとしてみるこ

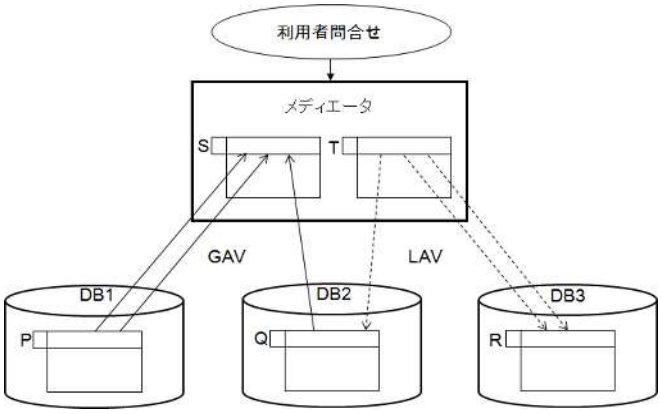

図 1　データ統合における GAV と LAV の関係

とができる．

グローバルスキーマに対する問合せからソースデータベースに対して問合せ処理を行う統合データベースシステムとしてメディエータを想定する．メディエータは異種の独立したソース(局所)データベースに対して，問合せのための利用者インタフェースを提供するデータ統合システムとする．

以下ではデータ統合及び演繹データベースの形式化について説明する．形式化と表記法は，文献[3],[4],[5]に従う．グローバルスキーマ R は，有限の関係 $\{R_{1,\ldots,}R_n\}$ と可能な無限の定義域 D から定義される．これらの関係記号，及び定数項となる D の要素とから一階述語論理 (First Order Logic, FOL) $L(R)$ が定義される．ビュー表は，グローバルの述語 V を用いて，$L(R)$ における式 $V(\bar{t}) \leftarrow body(\varphi v)$ で定義される．ここで $\bar{t}$ は，変数と定数項を含み，$body(\varphi v)$ は R の原子論理式(アトム)の連言である．この連言の問合せによって定義される問合せをグローバルスキーマへのビュー表とする．

スキーマ R のデータベースインスタンス D は，定義域 D を持つ FOL とし，整合性制約は言語 $L(R)$ で記述される FOL の式 φ とする．またインスタンス D が ψ を充足することを $D \vDash \psi$ とし，ψ が D で真となる場合とする．スキーマ R のデータベースインスタンス D，及び φv が与えられた場合，$\varphi v(D)$ は φv の定義を D に適用して得られる V の外延とする．

データベース言語として，論理型言語 Datalog[6] を用いる．Datalog の形式は次で与えられる．節は以下の式で表され，A_i は $i \in \{1, \ldots, \mathrm{n}\}$ とする原子論理式(アトム)である．また式に現れる変数は全称量化子∀で束縛される．

$$\forall A_1 \vee \ldots \vee A_k \vee \neg A_{k+1} \vee \ldots \vee \neg A_n$$

この式を同値な式で書き換え，全称量化子を省略した式を以下に示す．

$$A_1, \ldots, A_k \leftarrow A_{k+1}, \ldots, A_n$$

含意記号の左側の $(A_1, \ldots, A_k)$ をヘッド，右側の $(A_{k+1}, \ldots, A_n)$ をボディと呼ぶ．$k = n$ のときボディは空であり，$k = 0$ のとき，ヘッドは空である．確定節(Definite Clause)は一つのアトムを持つ $(k = 0)$．ヘッドが空の式は，問合せ(Query)である．また基底論理式は変数を持たない．問合せに対する結果は，データベースに対して結果の基底論理式が真となる問合せの論理式の基底代入である．これは，基底代入により基底論理式となった問合せは，データベースから論理的に含意されることを意味する．空のボディを持つ基底アトムをファクト(Fact)とも呼ぶ．演繹データベースは，外延データベース (Extensional Database, EDB), 内包データベース (Intentional Database, IDB)，整合性制約 (Integrity Constraint, IC) から構成される．EDB はファクトであり，IDB はルール(規則)の集まりである．

2.2. 統合データベースにおける整合性制約

ここでは，データ統合の方式と定式化について述べる．ここで，データベースの整合性について次のように定義する．

定義：データベース DB が無矛盾 (Consistent) であるとは，DB が IC を充足するとき，そのときに限る．これを次のように定義する．

$$DB \vDash IC$$

データ統合における整合性制約は，ソースデータベースでは整合性が保障されているが，統合データベースの元ではその保障は無い．GAV または LAV において，グローバルスキーマ上の整合性制約が成り立つことが仮定されるため，いずれにおいても違反の可能性がある．以下に，関数従属性 FD (Functional Dependency) が統合データベースのもとでは違反する例を示す．ここで，関数従属性は以下のルールとして定義されているものとする．

$$\forall_x \forall_y \forall_z ((s_1(x,y) \wedge s_2(x,z)) \rightarrow y = z$$

これは，Datalog では次のように記述される．

$$Y = Z \leftarrow S_1(X,Y), S_2(X,Z)$$

例．グローバルスキーマ上のビュー表 $R(X,Y)$ とソーススキーマ上の表 $\{V_1(a;b); V_2(c;d)\}$，及び $\{V_2(a;c); V_2(d;e)\}$ が次のように定義されている．

$$R(X,Y) \leftarrow S_1(X,Y)\ with\ S_1 = \{(a,b),(c,d)\}$$
$$R(X,Y) \leftarrow S_2(X,Y)\ with\ S_2 = \{(a,c),(d,e)\}$$

$S_1: \mathrm{X} \rightarrow \mathrm{Y}$ 及び $S_2: \mathrm{X} \rightarrow \mathrm{Y}$ は関数従属性に従うが，$R: \mathrm{X} \rightarrow \mathrm{Y}$ は関数従属性に違反する．データ統合においては，それぞれのソースデータベース内では整合性が維持されるが，データを統合することにより，グローバルスキーマの整合性制約によって違反が検出される．通常の二値論に基づく論理表現では，整合性制約を維持するために違反するデータの削除や更新によって整合性を保障し，データベースの修復を行うことで整合

性のあるデータベースベースの状態を維持する必要がある．

3. データ統合における Markov Logic Network の適用

本節では，Markov Logic Network とその処理系である Tuffy をデータ統合に用いるための仕組みについて説明する．

3.1. Markov Logic Network

Markov Logic Network (MLN) は，一階述語論理 (FOL) に Markov Network を組み合わせた言語であり，Markov Network を構築するためのテンプレート言語とみなすことができる[7]．MLN では，定数を項とする基底アトムの集合である可能世界の確率分布を定義する．可能世界を確率的に扱うことで，論理の矛盾を許容する表現を可能とする．FOL の式F_i に含まれる変数に定数を代入して得られる論理式を基底アトム (Ground Atom) という．MLN において，可能世界 x の確率分布は，基底アトムF_iとそれに対応する実数値の重みw_iの組の集合に対して次の式で与えられる．

$$\mathrm{P}(X = x) = \frac{1}{Z} exp\left(\sum_i w_i f_i(x)\right)$$

$f_i(x)$ は素性関数 $f_i(x) \in \{0,1\}$ であり，Z は正規化項である．MLN により違反を許容する論理式と，違反を許さない論理式を同時に扱うことが可能であり，MLN を用いてデータベースを記述することにより，確率を考慮した問合せの推論を行うことができる．

3.2. データ統合システムの実装

MLN によるデータ統合システムとして，その処理系である Tuffy [8],[9]を利用する．Tuffy では，推論処理に RDBMS を利用することで，推論処理の高速化と大量データへの対応を行っている．Tuffy のプログラムは，述語 (predicate) のリストと節形式によるルール (rule) の集合の 2 つの部分から成る．述語はスキーマと呼ばれ，述語と項の宣言を行う．先頭に* (star) が付加された述語は，閉世界仮説 (Closed World Assumption, CWA) により解釈される．ルールは基本的に Datalog と同じ節形式で記述されるが，Soft Rule と Hard Rule を指定することが異なる．Soft Rule は，重みを示す実数を先頭に付加する．Hard Rule は最大限の重みを持つルールであり，ピリオドを末尾に付加することで示す．

データ統合のモデルは，GAV によるメディエータシステムとし，Tuffy とメディエータシステムの構成要素の対応を定義する．データ統合のグローバルスキーマは，Tuffy のスキーマ，メディエータの実行エンジンを Tuffy のプログラムによる推論規則の集合とする．また，Tuffy の基底アトムの集合であるエビデンス (Evidence) を EDB，プログラムを IDB とする．グローバルスキーマに対する問合せは，Tuffy への問合せである．

以下では，Tuffy におけるデータ統合の適用方式について述べる．検討に用いるデータベースとして，ソースデータベース DB1, DB2 に水平分散されている社員情報データベースを仮定する．データ統合では，DB1 と DB2 を統合し，メディエータとしての MLN により，統合データベースに対する問合せを行う．

グローバルスキーマ．メディエータシステムにおけるグローバルスキーマを Tuffy のスキーマとして定義する．スキーマは，述語の定義であり，ルール，及び問合せで使用する．スキーマの例を以下に示す．

```
*EmpID(id, name)
*Salary(id, salary)
*Department(id, department)
EmpSalary(id, name, salary)
```

先頭に*が付加された述語は，閉世界仮説により解釈される．

EDB (外延データベース)．Tuffy のエビデンスを統合された EDB とみなし，統合されたソースデータベースがエビデンスとして登録された状態であるとする．従って，Tuffy のエビデンスは，統合された基底アトムの集合である．ソースデータベースの例を表 1 に示す．

表 1　ソースデータベースの例

DB1

EmpID	*id*	*name*
	1	A
	2	B
	3	C

Salary	*id*	*salary*
	1	50
	2	50
	3	50

DB2

EmpID	*id*	*name*
	1	A
	2	B
	4	D

Salary	*Id*	*salary*
	1	100
	2	80
	4	50

IDB (内包データベース)．問合せのための推論規則を記述する．関係 EmpSalary を導出するためのルールを FOL の形式で示す．

$$\forall x_1, x_2, x_3\ EmpID(x_1, x_2) \wedge Salary(x_1, x_3) \Rightarrow EmpSalary(x_1, x_2, x_3)$$

この式は，Tuffy では以下の節形式で表される．

$$!\,\mathrm{EmpID}(i1, n1) \vee !\,\mathrm{Salary}(i1, s1) \vee \mathrm{EmpSalary}(i1, n1, s1)$$

関係 EmpID と Salary からルールを用いて問合せである EmpSalary の結果を導出する．

IC (整合性制約)．グローバルスキーマに対する整合性制約を記述する．ここでは整合性制約として，関数従

属性を対象とする.FOL による IC の例を以下に示す.

$$\forall x_1, x_2, x_3, x_4 EmpID(x_1, x_2) \wedge EmpSalary(x_1, x_2, x_3) \wedge EmpSalary(x_1, x_2, x_4) \Longrightarrow x_3 = x_4$$

この式は，Tuffy の以下の節形式で表される.

$$!EmpID(x_1, x_2) \vee !EmpSalary(x_1, x_2, x_3) \vee !EmpSalary(x_1, x_2, x_4) \vee x_3 = x_4$$

これは，主キー制約であり，同一の EmpID を持つ社員の Salary は同じでなければならないことを意味する. 等式述語 (Equality Predicate) は，Tuffy の組み込み述語である．表 EmpID の同一のキー*id*に対して，表 EmpSalary の *salary* が一意にならない場合，等式が成り立たない.

IC は，Soft Rule，Hard Rule のいずれとしても定義することが可能であるが，制約の適用により，推論される結果に付与される確率に反映される．Hard Rule とすることで，エビデンスに存在しない基底アトムの導出を抑制することができる．Soft Rule では，エビデンスには存在しない定数項を持つ基底アトムの生成を許す．Soft Rule では重みを学習または任意に設定することができる．重みは大きいほど確度が高くなる.

学習．統合データベースに対して，学習によりルール及び整合性制約に対する重みを与える．訓練データとして，整合性のあるソースデータベースを用いるか，統合されたデータベースを用いるかで，学習結果の変動が予想される．学習用のデータとして，整合性のあるソースデータベースを用いることで，データの信頼性や優先などを学習させることを目的とするのであれば，正解とすべきソースデータベースを用いる．例えば，DB1 が DB2 より，データの重要度や信頼性が高い場合に，ルールとしていずれかのデータベースに含まれる定数項を含むルールにより高い重み付けを行うことや，導出される結果に高い重みを与えることが考えられる. 今回の実験では，重要性や信頼性の高いソースデータベースを学習に用いる方法を適用する.

問合せ．問合せは，Tuffy のスキーマで定義される述語を用いて，グローバルスキーマに対して行う．問合せの例を次に示す.

EmpSalary(id, name, salary)

また条件を指定する場合は以下のように記述する.

EmpSalary(id, "A", salary)

Tuffy の問合せは，問合せの述語をオプションにより指定することで実行するが，演繹データベースとして，問合せは以下のように解釈される.

EmpSalary(id, "A", salary) ⟶

4. 実験

以下に問合せ EmpSalary(id, name, salary) の結果例を示す．整合性制約として，EmpSalary に対する主キー制約を定義する.

ケース 1. EDB: 学習データとして表 1 の DB1 とその問合せ結果のデータを使用する．推論用データとして表 1 の DB1，DB2 をマージし，問合せ結果を削除したものを使用する.

スキーマ:

```
*EmpID(id, name)
*Salary(id, salary)
EmpSalary(id, name, salary)
```

IDB: 学習により重みが付加されたルールを使用する. Tuffy による重みを学習したルールの例を以下に示す.

4.7629 !EmpID(v0,v1) v !Salary(v0, v2) v EmpSalary(v0,v1,v2)

IC: ∅ (無し).

Query: EmpSalary(id, name, salary)

Result: Tuffy による推論の結果を以下に示す.

```
1.0000    EmpSalary(2, "B", 80)
1.0000    EmpSalary(1, "A", 100)
1.0000    EmpSalary(1, "A", 50)
0.9900    EmpSalary(2, "B", 50)
0.9900    EmpSalary(4, "D", 50)
0.9700    EmpSalary(3, "C", 50)
```

統合された全てのデータがほぼ同じ確度となる.

ケース 2. EDB: ケース 1 に同じ.

スキーマ: ケース 1 に同じ.

IDB: ケース 1 に同じ.

IC: Hard Rule として以下のルールを指定する.

!EmpID(i1, n1) v !EmpSalary(i1, n1, s1) v !EmpSalary(i1, n1, s2) v s1 = s2 .

Query: EmpSalary(id, name, salary)

Result: Tuffy による推論の結果を以下に示す.

```
1.0000    EmpSalary(3, "C", 50)
1.0000    EmpSalary(4, "D", 50)
0.5300    EmpSalary(2, "B", 80)
0.5100    EmpSalary(1, "A", 50)
0.4800    EmpSalary(1, "A", 100)
0.4700    EmpSalary(2, "B", 50)
```

キー項目"A"については，DB1 と DB2 で salary の値が異なっている．結果として，それぞれ 0.51 と 0.48 というスコアとなり，ほぼ同程度の確からしさであると解釈できる.

IC を適切に指定することにより，キー制約に違反する数だけ確率が減少すると予想されるが，違反するデータ同士を弁別する特徴を表現することはできない. データのある属性に対して順序付けるルールを作ることによって，整合性制約に違反した場合のデータの信頼の順序を表すルールを定義できれば，矛盾するデータの許容に加え，情報の保持と情報同士の信頼度を区別することが可能となる.

次のケース 3 では，関係 Salary が(id, salary, date)の 3 番目の項の日付(date)属性を持っていた場合，日付の値を比較し，より新しい日付を持つタプルを結論として導出するルールを作成し，日付の順序を区別した推論が行われることを検証する.

ケース 3. EDB: 関係 EmpID は表 1 と同じとし，DB1，DB2 を以下に示す.

DB1

Salary	*id*	*salary*	*date*
	1	50	0501
	2	50	0501
	3	50	0501

DB2

Salary	*id*	*salary*	*date*
	1	100	1120
	2	80	1120
	4	50	1120

スキーマ:

```
*EmpID(id, name)
*Salary(id, salary, date)
EmpSalary(id, name, salary, date)
```

IDB: 学習により重みが付加されたルールを使用する．Tuffy による重みを学習したルールの例を以下に示す．

4.7772 !EmpID(v0, v1) v !Salary(v0, v2, v3) v
EmpSalary(v0, v1, v2, v3)

IC: キー制約を Hard Rule として定義する．

!EmpID(v0, v1) v !EmpSalary(v0, v1, v2, v3) v
!EmpSalary(v0, v1, v4, v5) v [v2 = v4].

また date の前後関係がある場合に，date の値の大小を区別するために以下のルールを追加する．

$$\forall x_1,x_2,x_3,x_4 EmpID(x_1,x_2) \wedge EmpSalary(x_1,x_2,x_3,x_4) \wedge EmpSalary(x_1,x_2,x_5,x_6) \wedge Salary(x_1,x_3,x_4) \wedge Salary(x_1,x_5,x_6) \Longrightarrow [x_4 > x_6]$$

Tuffy による重みを学習したルールの例を以下に示す．

-4.7772 !EmpID(v0, v1) v !EmpSalary(v0, v1, v2, v3)
v !EmpSalary(v0, v1, v4, v5) v !Salary(v0, v2, v3) v
!Salary(v0, v4, v5) v [v3 > v5]

Query: EmpSalary(id, name, salary, date)

Result: Tuffy による推論の結果を以下に示す．

```
1.0000  EmpSalary(3, "C", 50, 501)
1.0000  EmpSalary(4, "D", 50, 1120)
0.5700  EmpSalary(1, "A", 100, 1120)
0.5600  EmpSalary(2, "B", 80, 1120)
0.4400  EmpSalary(2, "B", 50, 501)
0.4300  EmpSalary(1, "A", 50, 501)
```

DB1, DB2 の一方にのみ存在する”C”と“D”は，高い確率となる．また，DB1, DB2 それぞれに存在する”A”と”B”は，ケース 2 と同様な確率を示している．また加えて，日時の大小関係により，より大きな date の値を持つ値の方が小さな値の date よりやや高いスコアを示している．

5. 考察

ケース 1 及びケース 2 の結果より，データ統合において，整合性制約として関数従属性によるキー制約を指定することで，整合性制約に違反するデータの存在を許容することができた．また，MLN では全ての定数項と関係（述語）から存在しない基底アトムを生成するが，IC によって不要な基底アトムの生成を排除することができたと推察する．ケース 3 では，整合性制約に違反するデータの組みに対して，信頼度による順序の表現を，IC を拡張することにより試みた．その結果，データ量が少量ではあるが，拡張したルールに従ったスコアの順序を結果として導出することができた．

値の比較のためのルールは，整合性制約とは矛盾するものであるが，MLN の Hard Rule と Soft Rule を適切に用いることにより，矛盾を適切に管理することが可能である．今回の実験では，キー制約を Hard Rule とし，追加した比較のためのルールより強い制約とした．

ルールの重みは学習データの与え方により変動するため，結果データに適切なスコアを与えるためには，優先すべき結果のサンプルを用いるなどの考慮が必要である．

6. おわりに

本稿では，データ統合における整合性制約の違反について，データ統合メディエータへの MLN を利用した解決の可能性を検討した．MLN を用いて問合せを行う際に適用する整合性制約と，制約に違反した場合にデータの信頼性を確率によって表現するためのルールを設計し，整合性制約に違反したデータを許容する統合データベースの構築が可能であることを検証した．これによって，整合性制約の違反を許容することで情報を失うことなくデータ統合の構築が可能であることが確認できた．整合性と IDB のルールの関係を考慮して重みを付与することが結果に影響のあることが分かってきたが，整合性制約とルール間の関係を考慮して最適な重み付けを行うための手法は課題である．今後の方針として，これらの課題に取り組むとともに提案手法の実効性の検証を進める．

参 考 文 献

[1] J. D. Ullman, “Information Integration Using Logical Views”, In Database Theory ICDT '97 Lecture Notes in Computer Science Volume 1186, pp.19-40, 1997.

[2] M. Lenzerini, "Data integration: a theoretical perspective", In Proceedings of PODS'02, pp.233-246, 2002.

[3] L. Bertossi, ”Database Repairing and Consistent Query Answering”, Synthesis Lectures on Data Management, 2011.

[4] L. Bertossi, J. Chomicki, A. Cortés, and C. Gutiérrez, “Consistent Answers from Integrated Data Sources”, FQAS’02, Flexible Query Answering Systems, 2002.

[5] J. Grant and J. Minker, “A logic-based approach to data integration”, Theory and Practice of Logic Programming archive, Vol.2, pp.323-368, 2002.

[6] H. Garcia-Molina, J. D. Ullman, J. Wisdom, “Database Systems: the Complete Book”, Prentice Hall, pp.463-502, 2002.

[7] P. Domingos and D. Lowd, “Markov Logic: An Interface Layer for AI”, Morgan & Claypool, 2008.

[8] F. Niu, C. Ré, A. Doan, and J. Shavlik, “Tuffy: Scaling up Statistical Inference in Markov Logic Networks using an RDBMS”, In Journal Proceedings of the VLDB Endowment, pp.373-384, 2011.

[9] Tuffy, http://i.stanford.edu/hazy/hazy/tuffy/