Дедуктивный подход к эволюции схем слабоструктурированных данных

Д.В. Луцив
СПбГУ
dluciv@lanit-tercom.com

19 февраля 2005

О чем пойдет речь

1. Слабоструктурированные иерархические БД, специфика, схема
2. Инварианты: множества и дедукция
3. Дедукция: технические детали
4. Рефакторинг: операции
5. Дальнейшее развитие темы

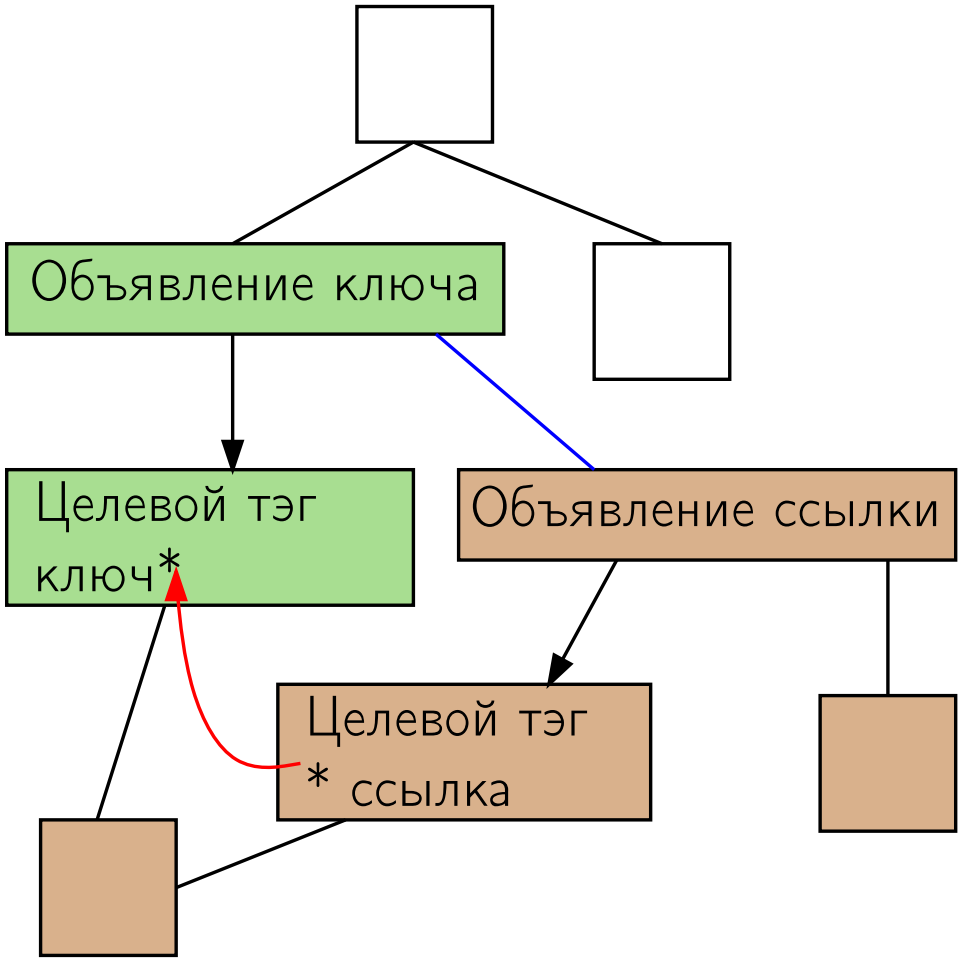
Специфика схем XML БД

- иерархическая схема
- схема допускает степени свободы
  - многие варианты содержимого сущностей
  - возможное отсутствие описания части схемы
- практически схема часто меняется
  - версионирование приложений и протоколов
  - различные проекции между XML и материальными данными

Инварианты: исходные

	Аксиома...	Вид	
(1)	замкнутости	$\forall t \in \tau,$	$P_e(t) \subset \tau$
(2)	ацикличности	$\forall t \in \tau,$	$t \notin \bigcup_{s \in \tau} \alpha_x(PL(x), P(s))$
(3)	общего корня “$\top$”axiom	$\exists \top \in \tau,$	$\forall t \in \tau \vert \top \in PL(t) \wedge P_e(\top) = \emptyset$
(4)	общего листа “$\bot$”	$\exists \bot \in \tau,$	$\forall t \in \tau \vert t \in PL(\bot)$
(5)	непосредственных предков	$\forall t \in \tau,$	$P(t) = P_e(t) \cup$ $\cup (P_e(t) \backslash \alpha_s(PL(s), P(t)))$
(6)	графа предков	$\forall t \in \tau,$	$PL(t) =$ $\bigcup_{s \in \tau} \alpha_x(PL(x), P(s)) \cup t$
(7)	интерфейса	$\forall t \in \tau,$	$I(t) = N(t) \cup H(t)$
(8)	собственных атрибутов	$\forall t \in \tau,$	$N(t) = N_e(t) \backslash H(t)$
(9)	унаследованных атрибутов	$\forall t \in \tau,$	$H(t) = \bigcup_{s \in \tau} \alpha_x(I(x), P(s))$

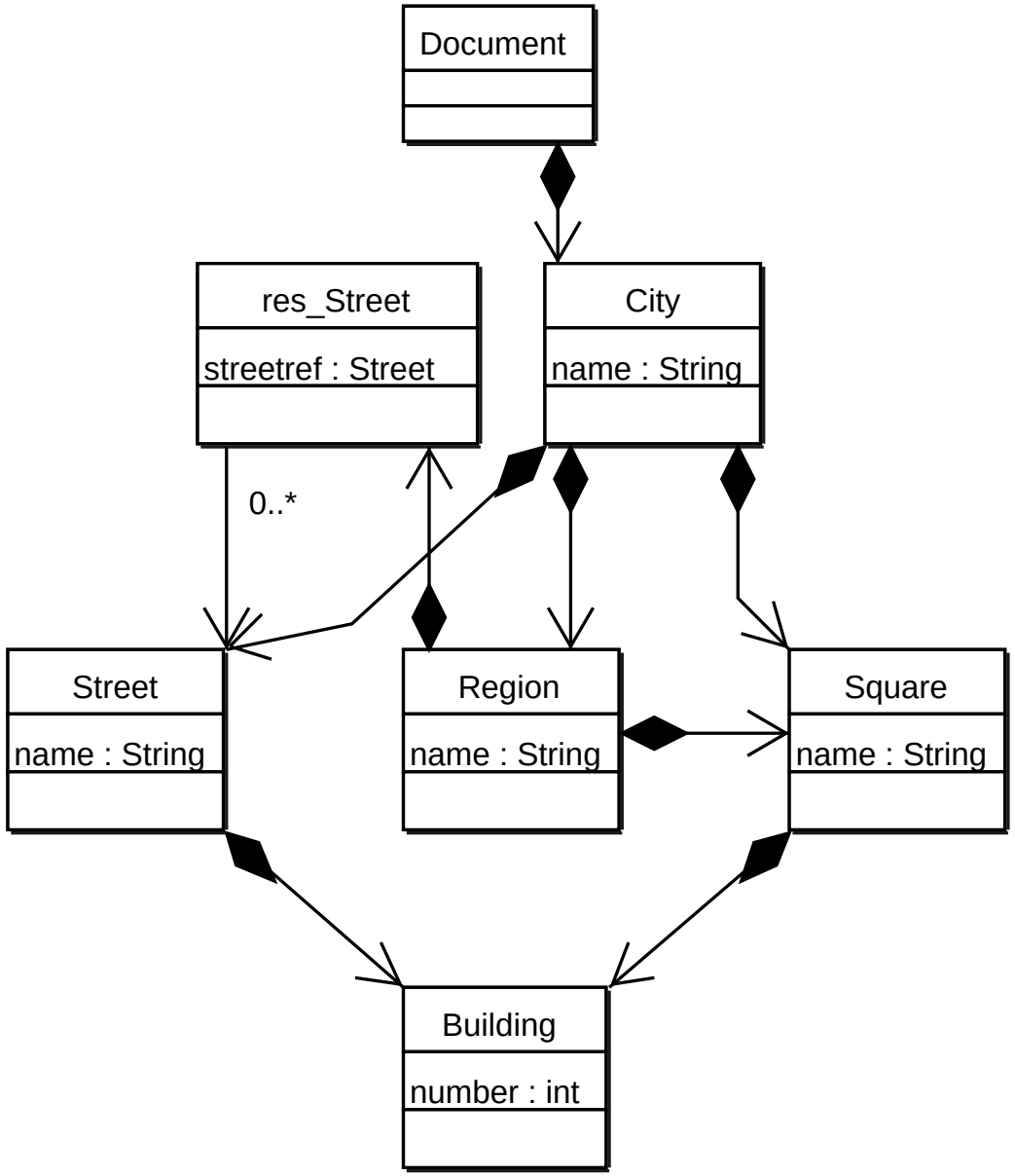
Покрытия


Объявление ключа
Целевой тэг
ключ*
Объявление ссылки
Целевой тэг
* ссылка

Инварианты: ключи и ссылки

Аксиома...	Вид
собственных ключей	$NK(t) = alpha_s(\mathsf{keydecl}($ $s, p : p(s) = t, a \in N(t)), PL(t))$
собственных ссылок	$NR(t) = \alpha_s(\mathsf{keyrefdecl}($ $s, p : p(s) = t, a \in N(t), *), PL(t))$
объявления ссылки	$\forall f_r \in R \quad \exists f_k \in K :$ $f_r = \mathsf{keyrefdecl}(t \in \tau, p \in SP,$ $m \in (A \cup \tau), f_k)$
покрытий	$\forall t \in \tau \quad CR(t) \in \alpha_s($ $\mathsf{keyrefdecl}(*, *, *, s), CK(t))$
существенных ссылок	$NR_e(t) = \alpha_s(\mathsf{keyrefdecl}($ $s, p : p(s) = t, a \in N_e(t)), PL(t), *)$

Модель


Document
res_Street
streetref : Street
City
name : String
0..*
Street
name : String
Region
name : String
Square
name : String
Building
number : int

Инварианты и факты:
как внушить машине факты

```
tag(':root'). tag(':term').

tag('Document'). tag('City'). tag('Square').
tag('Street'). tag('Region'). tag('res_Street').

inPe('City', 'Document'). inPe('Square', 'City').
inPe('Square', 'Region'). inPe('Street', 'City'). inPe('Region', 'City').
inPe('res_Street', 'Region'). inPe(':term', 'Square').
inPe(':term','Street').

inNe('City', 'City@name'). inNe('Street', 'Street@name').
inNe('Region', 'Region@name').
inNe('Square', 'Square@name'). inNe('res_Street', 'res_Street@streetref').

inNK('City', ['Street'], 'Street@name').
inNRe( 'City', ['Region', 'res_Street'], 'res_Street@streetref',
'City', ['Street'], 'Street@name' ).
```

Инварианты и факты:

как внушить машине инварианты

(5) Аксиома непосредственных предков $\forall t \in \tau, \quad P(t) = P_e(t) \cup \cup(P_e(t) \backslash \alpha_s(PL(s), P(t)))$

```
% Immediate Pred.
inP(T, S) :-
 inPe(T, S),
 inPe(T, X),
 X=\=S,
 not(inPL(X,S)).
```

```
inP(T, S) :-
 inPe(T, S),
 tag(X),
 X=\=S, %single Pe
 not(inPe(T,X)).
```

Инварианты и факты: пример того, что необходимо машине

```
% Immediate Predecessors
inP2(T, S) :-
 inPe(T, S),
 inPe(T, X),
 X=\=S,
 not(inPL(X,S)).

inP2(T, S) :-
 inPe(T, S),
 tag(X),
 X=\=S, %only one in Pe
 not(inPe(T,X)).

inP1(T,S) :-
 inP2(T,S),!.
```

```
inP(T,S) :-
 unbound(S),
 tag(S),
 inP1(T,S).

inP(T,S) :-
 unbound(T),
 tag(T),
 inP1(T,S).

inP(T,S) :-
 bound(T), bound(S),
 inP2(T,S),!.
```

Рефакторинг: операции (1)

- Объединение атрибутов, представляющих части целого
- Объединение тэгов
- Деление атрибута
- Деление тэга
- Добавление ключа
- Добавление атрибута для использования вместо ссылки

Рефакторинг: операции (2)

- Перемещение атрибута

  Дает возможность:

  – избежать разыменования для обращения к атрибуту другой сущности;

  – отделить редко используемые атрибуты для повышения быстродействия;

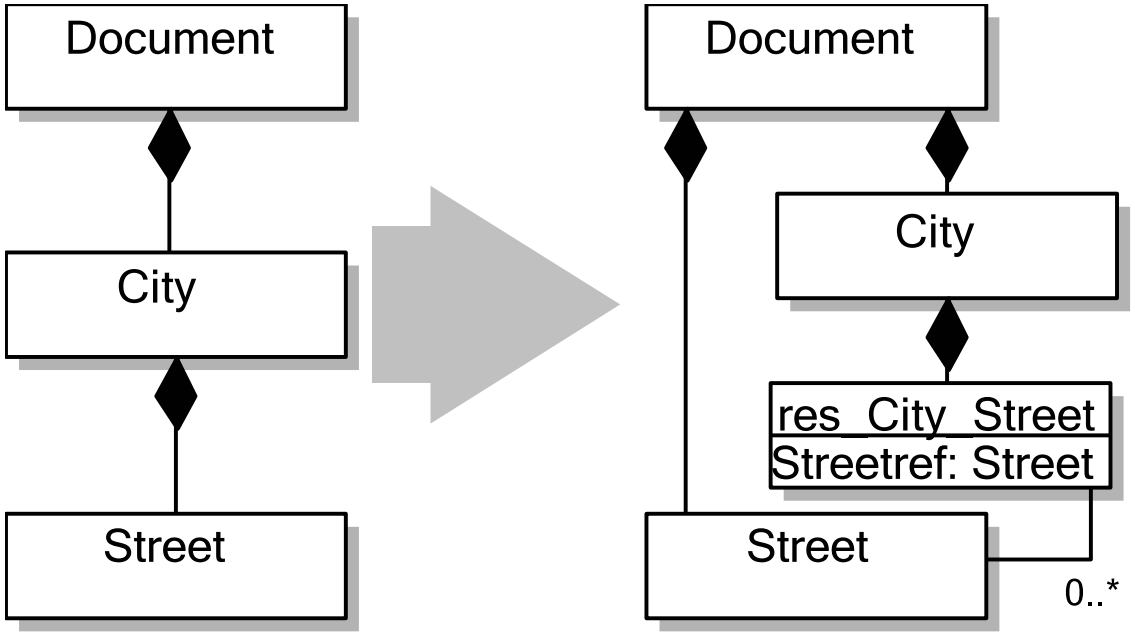
  – отделить доступные только для чтения атрибуты, если СУБД не контролирует доступ и в случаях, когда это поможет оптимизатору.

Рефакторинг: операции (3)

- Перемещение тэга в τ
- Безопасное удаление
- Замена естественного ключа сурогатным

Одна операция с примером

Замена агрегации “$1 \longrightarrow *$” на “$* \longrightarrow *$”



Рефакторинг: для более совершенных моделей

- Переименовать что-либо
- Распознать и восстановить для атрибутов стандартные типы
- Реализовать каскадное удаление для слабых сущностей
- Распознать и выделить в отдельные сущности группы схожих данных

Дальнейшее развитие

- Расширение схемы под конкретные задачи

- Сочетание дедуктивной БД с другими техниками, такими как системы переписывания фундаментальных структур, например:

  Termware
  http://www.gradsoft.com.ua/

- Продвижение модели в сторону онтологий

  – онтология - тоже модель данных;

  – позволяет интегрироваться с существующими системами, например:

  Prolog+CG http://prologpluscg.sourceforge.net/

  Amine Platform http://amine-platform.sourceforge.net/

Концепции: иерархия понятий

```
Universal > Person, Animal, Action, Situation,
 Object, AbstractEntity, Attribute.
Person > Man, Woman.
Man > Boy, Employee.
Woman > Girl, Employee.
Employee > Supervisor.
Action > Drive, Love, Break, Rent, Begin, Press, Look.
Object > Vehicle, Machine, Key, Keyboard, Finger.
AbstractEntity > Society, Session, Years, Proposition.
Vehicle > Car, Truck.
Attribute > Fast, Color, Expensive, Big.
```

Концепции: экземпляры

```
Boy = John, Bob, Sam, Andre.
Girl = Sue, Mary.
Color = red.
Machine = res23.
Years = four.
Key = enter.
```

Концептуальные графы: пример

Нотации близки к предложенным John F. Sowa http://www.jfsowa.com/
примеры:

```
[Man], [Man : John], [Man : {John, Carl, Henry}], [Cat : x],
[Integer = 25], [List = (1, 2, 3)],
[Date :CurrentDate = (04,01,2000)], [Term = papa(x, Hicham)],

 [Proposition :propHenry  = [Man : Carl]<-agnt-[Think]-obj
->
 [Proposition =[Man: Carl]-attr->[Crazy] ] ]
```

Спасибо