【VLDB2011勉強会】

Session 7:Query Processing

担当：成田 和世（NEC）

Generating Efficient Execution Plans for Vertically Partitioned XML Databases

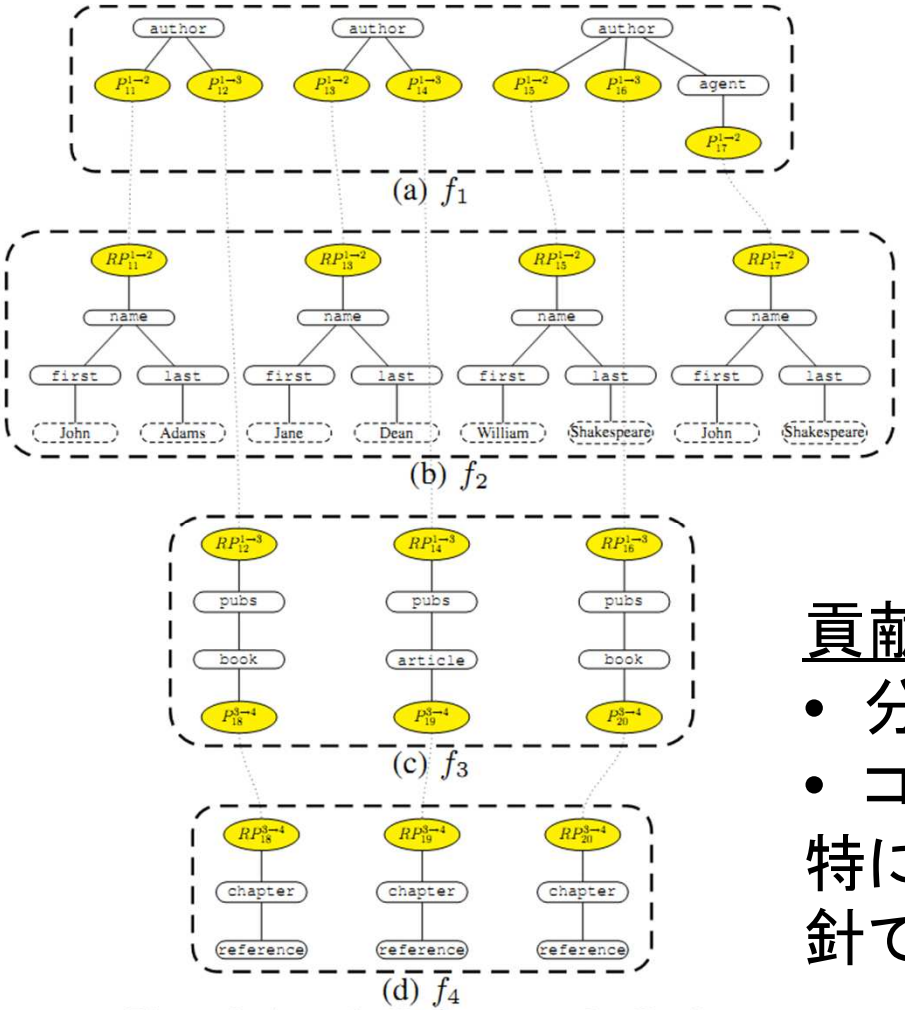


Figure 1: A vertically fragmented collection

論文の焦点：
分散XML DB（vertical 分割）に対する問い合わせ最適化

クエリq：

```
/author[name/last[.='Shakespeare']]//book//reference
```

<u>貢献</u>

- 分散実行木の最適化
- コスト推定

特にクエリの応答時間を最小化する方針で、分散実行計画を最適化している

図の引用 http://www.vldb.org/pvldb/vol4/p1-kling.pdf

提案

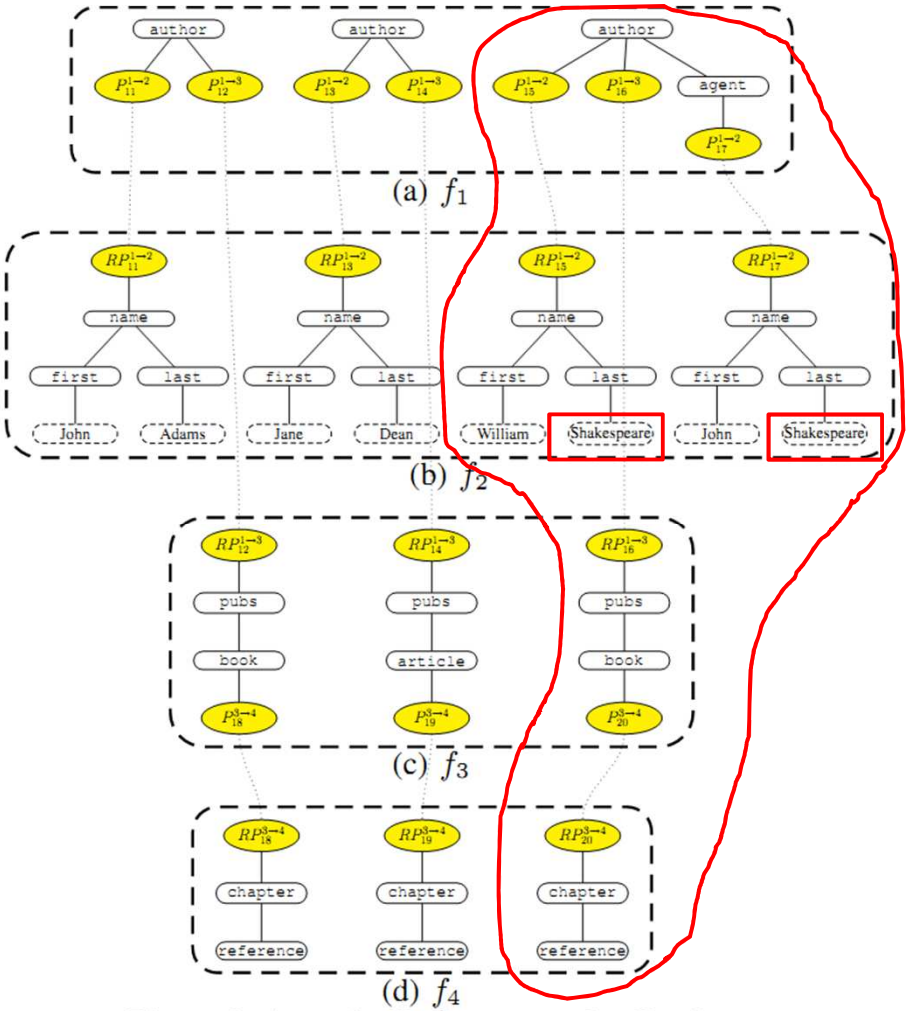


Figure 1: A vertically fragmented collection

クエリq:

```
/author[name/last[.='Shakespeare']]//book//reference
```

<u>分散実行木の構築</u>

サイト f1～f4 の局所plan (p1～p4) の評価結果をサイト間結合し、分散実行木を構築していく

※piの最適化法については別の論文を参照

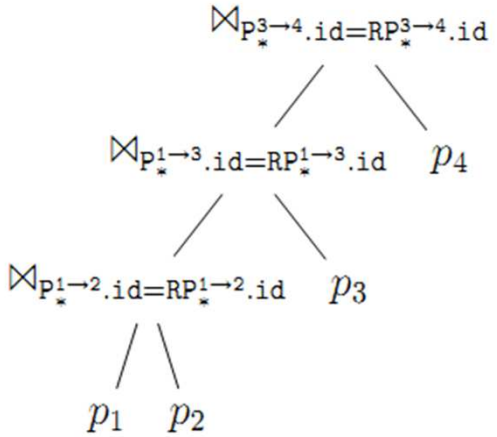
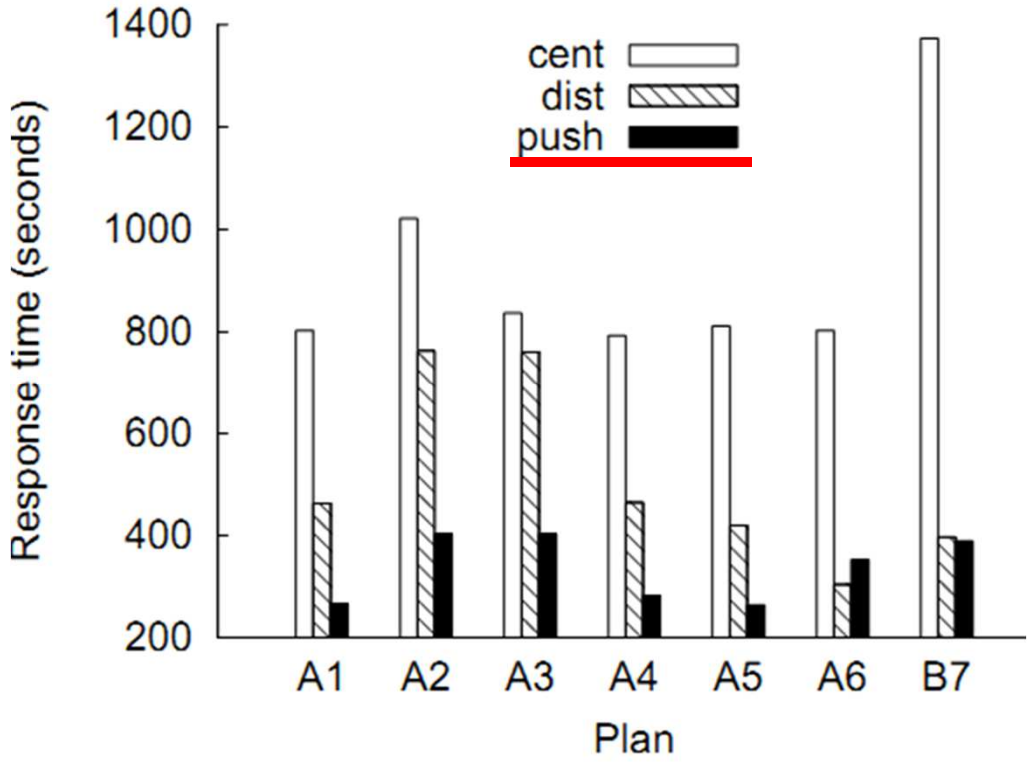


Figure 2: Distributed execution plan

図の引用 http://www.vldb.org/pvldb/vol4/p1-kling.pdf

評価

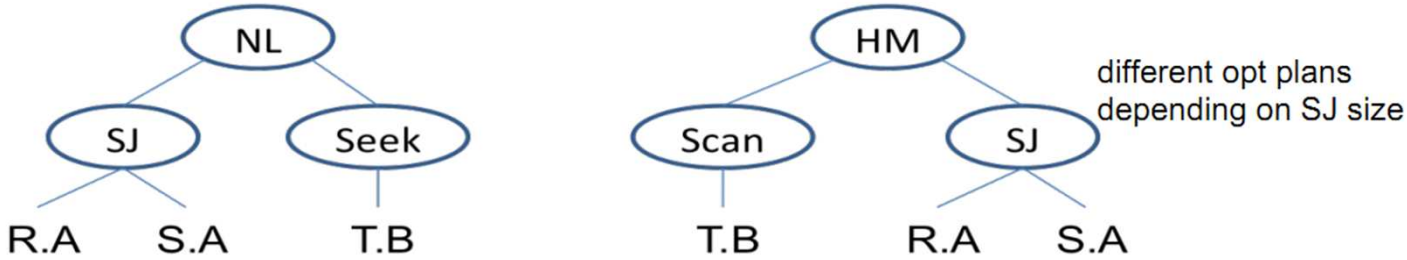
- XPathMarkベンチマーク
- cent･･･集中処理、dist･･･ナイーブな分散実行(left-deep)



図の引用 http://www.vldb.org/pvldb/vol4/p1-kling.pdf

Similarity Join Size Estimation using Locality Sensitive Hashing

- 背景:
  - 類似結合をDBMS上で活用したい
    - 問い合わせ最適化で適切なプランを出力するために、類似結合における結合サイズの推定が必要



  - 既存の単純なサンプリング法では、selectivityが小さいと結合サイズの見積りが甘くなる

τ	0.1	0.3	0.5	0.7	0.9
join size	105B	267M	11M	103K	42K
selectivity	33%	.085%	.0086%	.000064%	.000013%

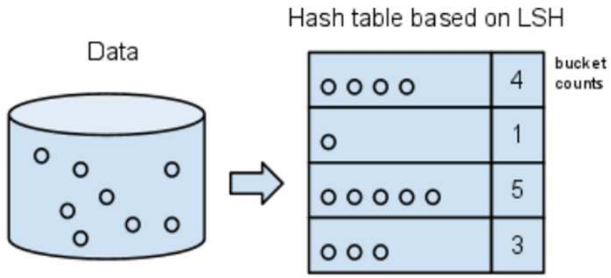
DBLP 800K

高い閾値(τ)の指定が一般的
※cosine similarity の場合

図の引用http://www.vldb.org/2011/files/slides/research7/rSession7-2.pdf

図の引用http://www.vldb.org/2011/files/slides/research7/rSession7-2.pdf
http://arxiv.org/ftp/arxiv/papers/1104/1104.3212.pdf

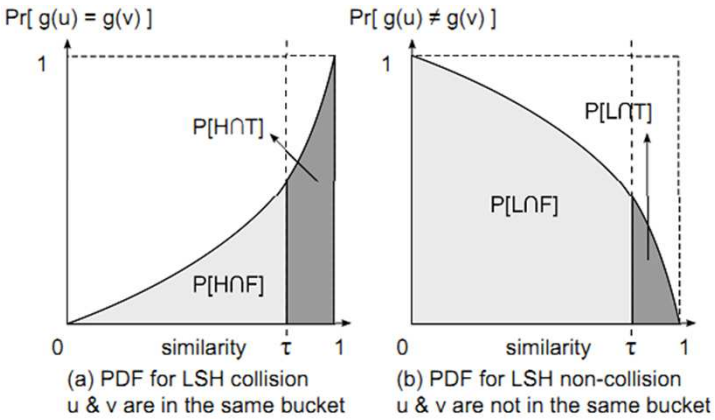
提案



- LSHを利用
  - LSH: 似ているオブジェクトに同じハッシュ値を割り当てる
- 類似オブジェクトを、LSHにより1つのバケットに振分
- カウント数（バケットサイズ）N_Hから結合サイズN_Tを推定

$$N_H = N_T * P(H|T) + N_F * P(H|F)$$

N_T：類似ペア数、P(H|T): 類似ペアが同じバケットに振り分けられる確率



条件つき確率を、LSHの性質に基づき推定する

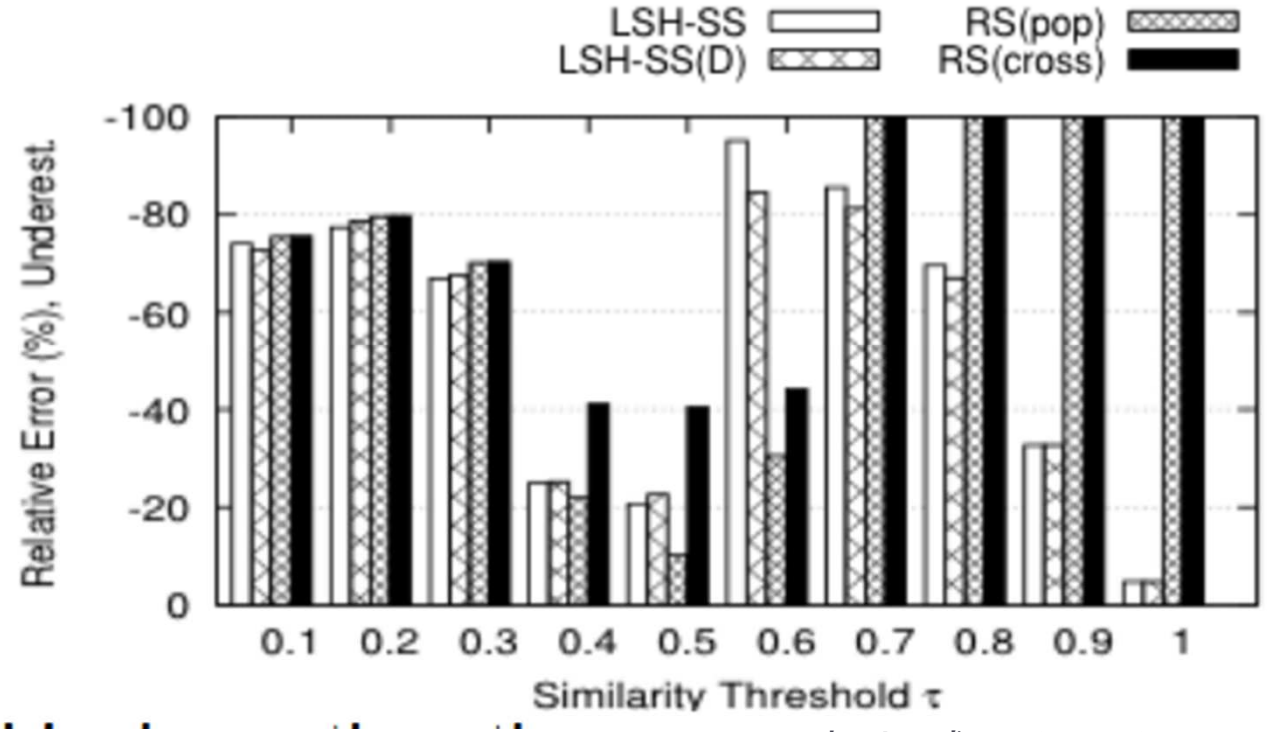
重要な性質：P[h(u)=h(v)]=sim(u,v)

- LSHのハッシュ関数群{h}が決まれば、確率密度関数（左図）が決まる。
- 確率密度関数が決まれば、条件つき確率が分かる。

図の引用 http://arxiv.org/ftp/arxiv/papers/1104/1104.3212.pdf

評価

- ランダムサンプリング（RS）と提案手法LSH-SSの精度比較
  - 縦軸＝推定サイズ/実際のサイズ
- 高い閾値に対してよい精度を示している



Accelerating Queries with Group-By and Join by Groupjoin

- 背景
  - 多くの集計クエリは、結合とGroupByがセット

Select **a**, count(*) as **s**, sum(**c**) as **t**
From **R** left outer join **S**
Group by **a**

R

a
1
2
3
3

S

b	c
2	2
2	3
3	4
3	5
3	6
4	7

R⟕S

a	b	c
1		
2	2	2
2	2	3
3	3	4
3	3	5
3	3	6
3	3	4
3	3	5
3	3	6

$\Gamma_{s:count(*);t:sum(c)}(R⟕S)$

a	s	t
1	0	null
2	2	5
3	3	15
3	3	15

- 結合とGroupByを一つのオペレータとしてマージ（groupjoin）すれば、実行時間が短縮できる

図の引用 http://www.vldb.org/2011/files/slides/research7/rSession7-3.pdf

図の引用 http://www.vldb.org/2011/files/slides/research7/rSession7-3.pdf

提案

- 1つのハッシュ表で結合とGroupオペレータを実施

Select **a**, count(*) as **s**, sum(**c**) as **t**
From **R** left outer join **S**
Group by **a**

R

a
1
2
3
3

S

b	c
2	2
2	3
3	4
3	5
3	6
4	7

a	s : count(*)	t : sum(c)
1	0	null
2	0	null
3	0	null

1．ハッシュ表をRと集計条件で初期化

a	s : count(*)	t : sum(c)
1	0	null
2	2	5
3	3	15

2．Sに関してハッシュ表をProbe

3．結果を出力

a	s	t
1	0	null
2	2	5
3	3	15
3	3	15

$$\mathrm{Cost}(R \bowtie S) = a + b|R| + c|S|$$

JoinとGroupingのシーケンスをgroupjoinに変換する代数表現を提供している

評価

- TPC-H

Query	time(ms) with ⋈	time(ms) old
9	192	192
21	127	500
13	84	278
3	70	104
10	51	74
5	59	68
16	45	49
20	37	37
17	33	34
...	...	...
total	1295	1932

図の引用 http://www.vldb.org/2011/files/slides/research7/rSession7-3.pdf

Optimizing Query Answering under Ontological Constraints

図の引用 http://www.vldb.org/2011/files/slides/research7/rSession7-4.pdf

- 背景
  - オントロジDB（D ∪ Σ）に対する問い合わせ

Database: $D = \{person(john)\}$ Johnは人間である

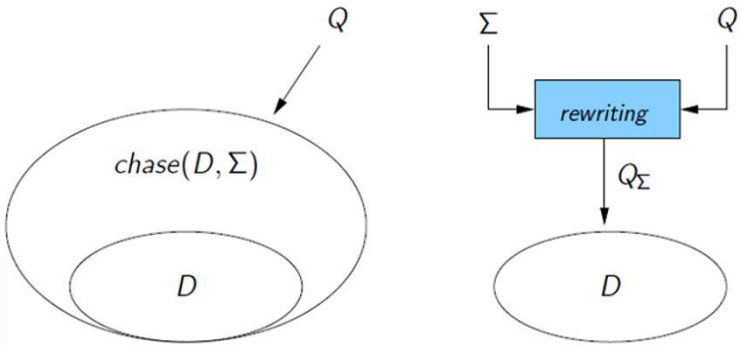
オントロジ理論Σ: $person(X) \rightarrow \exists Y\, father(Y, X), person(Y)$

全ての人間(X)の父親(Y)は人間である

結合クエリ(CQ): $q(B) \leftarrow father(A, B)$ 父親を持つ任意のBを出力せよ

  - 空間D ∪ Σを探索するのは大変なので、クエリをΣに基づきfirst-order queryにrewriteし、Dを探索する

書き換えクエリ: $q(B) \leftarrow person(B)$



Q_Σは膨大⇒コンパクトなQ_ΣのRewriting法を提案

評価

図の引用 http://www.vldb.org/2011/files/slides/research7/rSession7-4.pdf

- Algorithmは追い切れず…　m(_ _)m
- Rewritingにより生成されたfirst-order queryの数を、既存手法と比較

Ontology	Query	Size				
		QO	RQ	NY	PR	NY^{DTG}
A	Q_1	783	402	249	69	58
	Q_2	1812	103	94	52	41
	Q_3	4763	104	104	55	43
	Q_4	7251	492	456	93	81
	Q_5	66068	624	624	71	65
U	Q_1	5	2	2	6	5
	Q_2	287	148	1	1	1
	Q_3	1260	224	4	8	7
	Q_4	5364	1628	2	6	5
	Q_5	9245	2960	2	11	10
S	Q_1	6	6	6	7	7
	Q_2	204	160	1	3	3
	Q_3	1194	480	2	5	4
	Q_4	1632	960	2	5	4
	Q_5	11487	2880	4	7	6
P5X	Q_1	14	14	10	11	11
	Q_2	86	77	57	16	16
	Q_3	530	390	324	16	16
	Q_4	3,476	1,953	1,642	16	16
	Q_5	23,744	9,766	8,219	16	16

提案