

Tiresias

The Database Oracle for How-To Queries

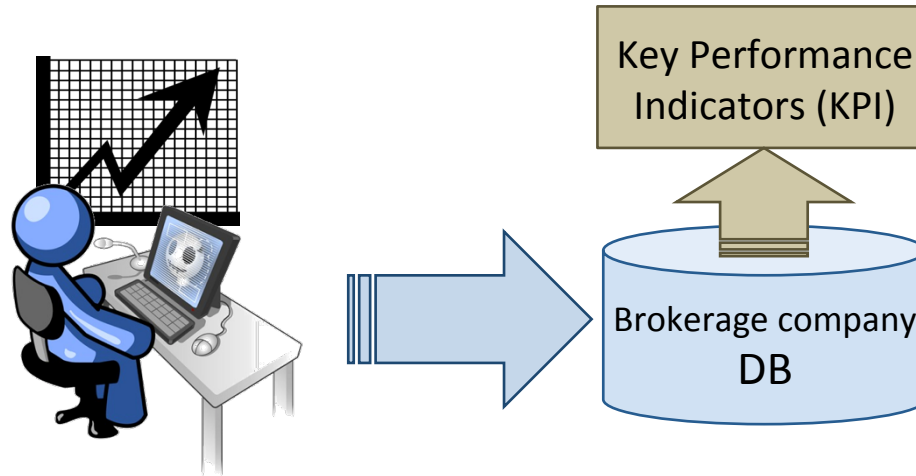
Alexandra Meliou[§]✦

Dan Suciu[✦]

[§]University of Massachusetts Amherst

✦University of Washington

Hypothetical (What-if) Queries

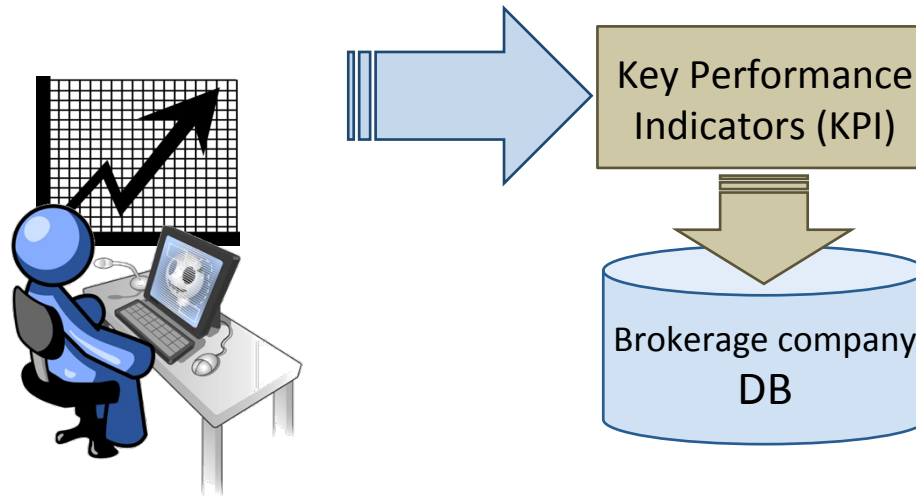


Example from [Balmin et al. VLDB'00]:

“An analyst of a brokerage company wants to know **what** would be the effect on the return of customers’ portfolios **if** during the last 3 years they had suggested Intel stocks instead of Motorola.”



How-To Queries



Modified example:

“An analyst wants to ask **how to** achieve a 10% return in customer portfolios, with the least number of trades.”



TPC-H example

- A manufacturing company keeps records of inventory orders in a LinetItem table.

⊙ KPI: Cannot order more than 7% of the inventory from any single country

(variables)

⊙ Can reassign orders to new suppliers as long as the supplier can supply the part

← (constraints)

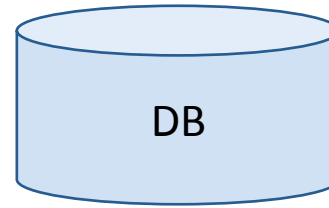
⊙ Minimize the number of changes

(optimization objective)

constraint optimization

Constraint Optimization on Big Data

this is for a set of
10 lineitems and 40
suppliers



Impractical!

```
example10.mod
#
# Example problem
#
/* sets */
set CORE_LineItem;
set CORE_Q;
set LineItem;
set Position;
set KEY_Q;
set KEY_Q_1;
set CORE_S;
set CORE_O;

/* parameters */
param join_CORE_LineItem_CORE_Q {i in CORE_LineItem} :
param LineItemQuant {i in LineItem} :
param join_CORE_Q_LineItem {i in CORE_Q} :
param join_CORE_Q_Position {i in CORE_Q} :
param key_CORE_Q {i in KEY_Q, j in CORE_Q} :
param key_CORE_Q_1 {i in KEY_Q_1, j in CORE_Q} :
param set_CORE_S_CORE_LineItem {i in CORE_S, j in CORE_LineItem} :

/* variables */
var CORE_Qid {i in CORE_Q} binary :
var varKEY_Q_0 {i in KEY_Q_0} binary := 1;
var varKEY_Q_1 {i in KEY_Q_1} binary := 1;
var CORE_Qquant {i in CORE_Q} :
var CORE_LineItemQuant {i in CORE_LineItem} binary :
var CORE_LineItemQuant {i in CORE_LineItem} :
var CORE_Sid {i in CORE_S} binary :
var CORE_Sc {i in CORE_S} :
var opt_CORE_LineItemQuant {i in CORE_LineItem} >= 0 :
var CORE_Obj1 {i in CORE_O} binary :
var CORE_Obj2 {i in CORE_O} :

/* constraints */
s.t. C1 {i in KEY_Q_0}: 1 >= sum{ j in CORE_Q} key_CORE_Q_0{j} * CORE_Qid{j};
s.t. C2 {i in KEY_Q_1}: 1 >= sum{ j in CORE_Q} key_CORE_Q_1{j} * CORE_Qid{j};
s.t. C3 {i in KEY_Q_0}: 1 <= sum{ j in CORE_Q} key_CORE_Q_0{j} * CORE_Qid{j};
s.t. C4 {i in CORE_Q}: CORE_Qquant[i] = LineItemQuant[ join_CORE_Q_LineItem[i] ];
s.t. C5 {i in CORE_LineItem}: CORE_LineItemQuant[i] = CORE_Qid[ join_CORE_LineItem_CORE_Q[i] ];
s.t. C6 {i in CORE_LineItem}: CORE_LineItemQuant[i] <= CORE_Qid[ join_CORE_LineItem_CORE_Q[i] ] *
CORE_LineItem[ join_CORE_LineItem_CORE_Q[i] ] * 666;
s.t. C7 {i in CORE_S, j in CORE_LineItem}: CORE_Sid[i] >= set_CORE_S_CORE_LineItem[i,j] *
CORE_LineItem[ join_CORE_LineItem_CORE_Q[i] ] * 666;
s.t. C8 {i in CORE_LineItem}: opt_CORE_LineItemQuant[i] <= CORE_LineItemQuant[i];
s.t. C9 {i in CORE_LineItem}: opt_CORE_LineItemQuant[i] <= CORE_LineItemQuant[i] * 666;
s.t. C10 {i in CORE_S}: CORE_Sc[i] = sum{ j in CORE_LineItem} set_CORE_S_CORE_LineItem[i,j] * (1 -
CORE_LineItem[ join_CORE_LineItem_CORE_Q[i] ] * 666);
s.t. C11 {i in CORE_S}: CORE_Sc[i] <= 50;
s.t. C12 {i in CORE_S}: CORE_Sc[i] >= 50;
s.t. C13 {i in CORE_O}: CORE_Obj1[i] <= sum{ j in CORE_Q} CORE_Qid[j];
s.t. C14 {i in CORE_O, j in CORE_Q}: CORE_Obj2[i] <= CORE_Qid[j];
s.t. C15 {i in CORE_O}: CORE_Obj2[i] = sum{ j in CORE_Q} CORE_Qid[j];

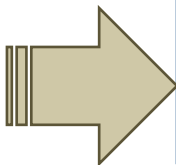
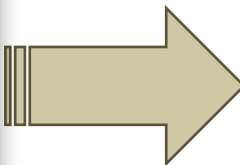
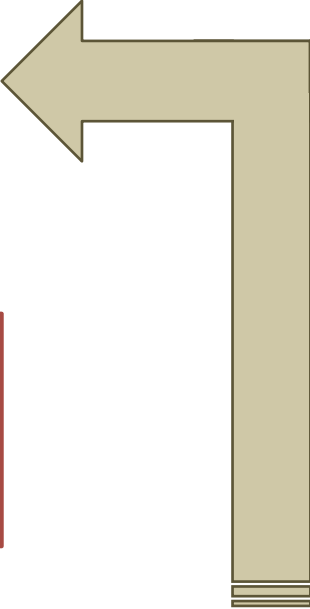
/* objective */
maximize z: sum{ i in CORE_O} (CORE_Obj1[i]);

solve;
print "n";
print "*****";
print "CORE_LineItem";
print "id quant";
print {i in CORE_LineItem} "x1 x1Val", 1, CORE_LineItem[i], CORE_LineItemQuant[i];
print "ObjVal";
print "*****";
print "n";
data;

set CORE_LineItem := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 ;
set CORE_Q := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 ;
set LineItem := 1 2 3 4 5 6 7 8 9 10 ;
set Position := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 ;
set KEY_Q_0 := 1 2 3 4 5 6 7 8 9 10 ;
set KEY_Q_1 := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 ;
set CORE_S := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 ;
set CORE_O := 1 ;
param join_CORE_LineItem_CORE_Q :=
1 10
2 15
3 14
4 13
5 24
6 23
7 23
8 23
9 23
10 23
```

Mixed Integer
Programming (MIP)
solver

transform
into data
updates



Demo: **Tiresias**

a tool that makes
how-to queries practical

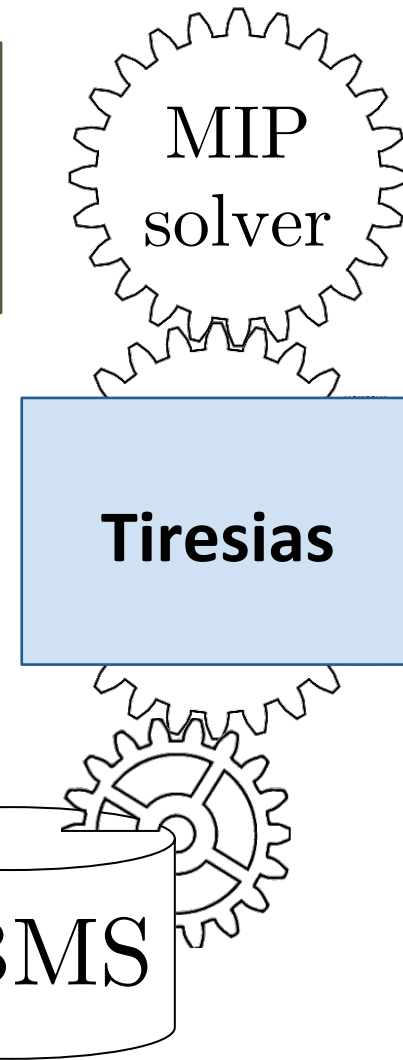
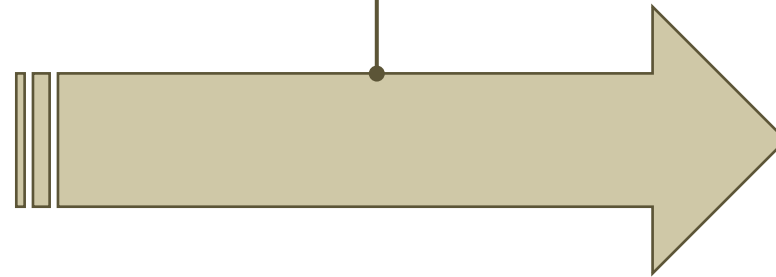
Tiresias: How-To Query Engine

TiQL (Tiresias Query Language)

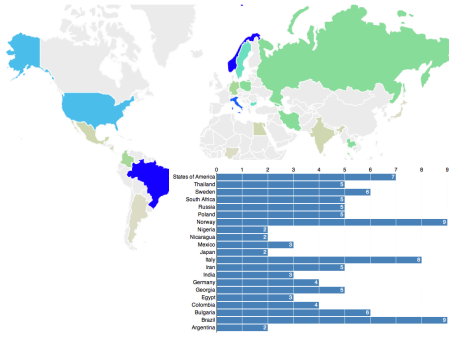
```
RULES:  
HChooseS(ok,pk,sk,qnt,sk',country) :- PartSupp(pk,sk') & LineItem(ok,pk,sk,qnt)  
                                     & SuppNation(sk2,country)  
HLineItem(ok,pk,sk',qnt)             :- HChooseS(ok,pk,sk,qnt,sk',country)  
HOrderSum(country,count(*))         :- HChooseS(ok,pk,sk,qnt,sk',country)  
[c? <= 10]                           <- HOrderSum(country,c?)  
MAXIMIZE(count(*)) :- HChooseS(ok,pk,sk,qnt,sk,country)
```



Declarative interface,
extension to Datalog



Overview



Visualizations

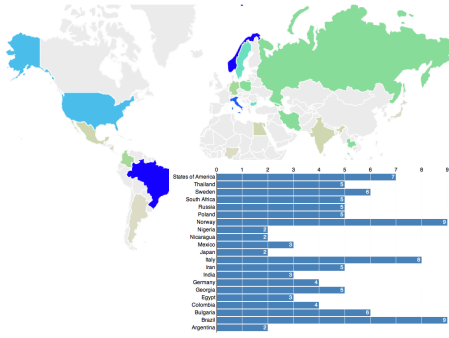
RULES:
HChooseS(ok, pk, sk, qnt, sk', country) :- PartSupp(pk, sk', country) & LineItem(ok, pk, sk, qnt, country) & SuppNation(sk2, country)
HLineItem(ok, pk, sk', qnt) :- HChooseS(ok, pk, sk, qnt, sk', country)
HOrderSum(country, count(*)) :- HChooseS(ok, pk, sk, qnt, sk', country)
[c? <= 10] <- HOrderSum(country, c?)
MAXIMIZE(count(*)) :- HChooseS(ok, pk, sk, qnt, sk, country)

TiQL

```
example10.mod
1 #
2 # Example problem
3 #
4
5 /* sets */
6 set CORE_LineItem;
7 set CORE_Q;
8 set LineItemB;
9 set PartSupp;
10 set KEY_Q_0;
11 set KEY_Q_1;
12 set CORE_S;
13 set CORE_QB;
14
15 /* parameters */
16 param join_CORE_LineItem2_CORE_Q {i in CORE_LineItem} :
17   param LineItemBqant {i in LineItemB} :
18   param join_CORE_Q_LineItemB {i in CORE_Q} :
19   param join_CORE_Q_PartSupp {i in CORE_Q} :
20   param key_CORE_Q_0 {i in KEY_Q_0, j in CORE_Q} :
21   param key_CORE_Q_1 {i in KEY_Q_1, j in CORE_Q} :
22   param ord_CORE_S_CORE_LineItem {i in CORE_S, j in CORE_LineItem} :
23
24 /* variables */
25 var CORE_QB {i in CORE_Q} binary ;
26 var varKEY_Q_0 {i in KEY_Q_0} binary s1;
27 var varKEY_Q_1 {i in KEY_Q_1} binary s1;
28 var CORE_QBqant {i in CORE_Q} :
29 var CORE_LineItemB2 {i in CORE_LineItem} binary ;
30 var CORE_LineItemBqant {i in CORE_LineItem} :
31 var CORE_QB {i in CORE_S} :
32 var CORE_S {i in CORE_S} :
33 var ord_CORE_LineItemBqant {i in CORE_LineItem} >= 0 :
34 var CORE_QB2 {i in CORE_QB} binary ;
35 var CORE_QB {i in CORE_QB} :
36
37 /* constraints */
38 s.t. CORE_Q_0 {i in KEY_Q_0} : 1 >= sum {j in CORE_Q} key_CORE_Q_0[i, j] * CORE_QB[j];
39 s.t. CORE_Q_1 {i in KEY_Q_1} : 1 >= sum {j in CORE_Q} key_CORE_Q_1[i, j] * CORE_QB[j];
40 s.t. CORE_Q_0 {i in KEY_Q_0} : 1 <= sum {j in CORE_Q} key_CORE_Q_0[i, j] * CORE_QB[j];
41 s.t. CORE_Q_1 {i in KEY_Q_1} : 1 <= sum {j in CORE_Q} key_CORE_Q_1[i, j] * CORE_QB[j];
42 s.t. CORE_Q {i in CORE_Q} : CORE_QBqant[i] = LineItemBqant[join_CORE_Q_LineItemB[i]];
43 s.t. CORE_LineItem {i in CORE_LineItem} : CORE_LineItemB[i] = CORE_QB[join_CORE_LineItem2_CORE_Q[i]];
44 s.t. CORE_LineItem {i in CORE_LineItem} : CORE_LineItemBqant[i] <= CORE_QB[join_CORE_LineItem2_CORE_Q[i]] * 666;
45 s.t. CORE_LineItem {i in CORE_LineItem} : CORE_LineItemBqant[i] >= CORE_QB[join_CORE_LineItem2_CORE_Q[i]] * 666;
46 s.t. CORE_LineItem {i in CORE_S} : CORE_QB[i] <= sum {j in CORE_LineItem} ord_CORE_S_LineItem[j, i] * CORE_LineItemB[j];
47 s.t. CORE_LineItem {i in CORE_S, j in CORE_LineItem} : CORE_QB[i] <= ord_CORE_S_LineItem[j, i] * CORE_LineItemB[j];
48 s.t. CORE_LineItem {i in CORE_LineItem} : ord_CORE_LineItemBqant[i] <= CORE_LineItemBqant[i];
49 s.t. CORE_LineItem {i in CORE_LineItem} : ord_CORE_LineItemBqant[i] <= CORE_LineItemBqant[i] * 666;
```

MathProg or AMPL

Overview



Visualizations

Demo

```
RULES:
HChooseS(ok,pk,sk,qnt,sk',country) :- PartSupp(pk,sk') & LineItem(ok,pk,sk,qnt)
& SuppNatioN(sk2,country)
HLineItem(ok,pk,sk',qnt) :- HChooseS(ok,pk,sk,qnt,sk',country)
HOrderSum(country,count(*) ) :- HChooseS(ok,pk,sk,qnt,sk',country)
[c? <= 10] <- HOrderSum(country,c?)
MAXIMIZE(count(*) ) :- HChooseS(ok,pk,sk,qnt,sk,country)
```

TiQL

```
example10.mod
#
# Example problem
#
/* sets */
set CORE_LineItem;
set CORE_Q;
set LineItemQ;
set PartSuppQ;
set KEY_Q_W;
set KEY_Q_W;
set CORE_S;
set CORE_QW;

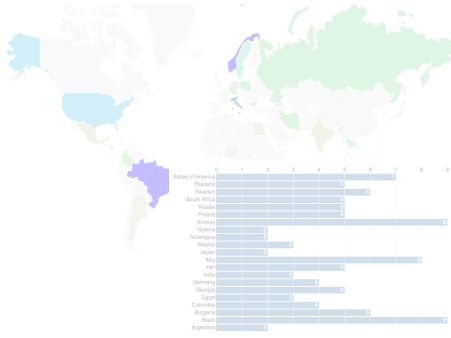
/* parameters */
param join_CORE_LineItem_CORE_Q {i in CORE_LineItem} ;
param LineItemSupp {i in LineItem} ;
param join_CORE_Q_LineItem {i in CORE_Q} ;
param join_CORE_Q_PartSupp {i in CORE_Q} ;
param key_CORE_Q {i in KEY_Q_W} in CORE_Q ;
param key_CORE_Q_1 {i in KEY_Q_1} in CORE_Q ;
param set_CORE_Q_CORE_LineItem {i in CORE_Q} in CORE_LineItem ;

/* variables */
var CORE_Q {i in CORE_Q} binary ;
var varKEY_Q_W {i in KEY_Q_W} binary x1;
var varKEY_Q_1 {i in KEY_Q_1} binary x1;
var CORE_Qsum {i in CORE_Q} ;
var CORE_LineItemQ {i in CORE_LineItem} binary ;
var CORE_LineItemQsum {i in CORE_LineItem} ;
var CORE_Q1 {i in CORE_Q} ;
var objv_CORE_LineItemQsum {i in CORE_LineItem} == 0 ;
var CORE_Q1Q {i in CORE_Q} binary ;
var CORE_Q1Q1 {i in CORE_Q} ;

/* constraints */
s.t. C1 {i in KEY_Q_W: 1 == sum{ j in CORE_Q} key_CORE_Q_W(j) * CORE_Q1(j)};
s.t. C2 {i in KEY_Q_1: 1 == sum{ j in CORE_Q} key_CORE_Q_1(j) * CORE_Q1(j)};
s.t. C3 {i in KEY_Q_W: 1 == sum{ j in CORE_Q} key_CORE_Q_W(j) * CORE_Q1(j)};
s.t. C4 {i in CORE_Q: CORE_Qsum(i) = LineItemSupp(join_CORE_Q_LineItem(i))};
s.t. C5 {i in CORE_LineItem: CORE_LineItemQ(i) = CORE_Q1Q(join_CORE_LineItem_CORE_Q(i))};
s.t. C6 {i in CORE_LineItem: CORE_LineItemQsum(i) = CORE_Q1Q1(join_CORE_LineItem_CORE_Q(i)) * 666};
s.t. C7 {i in CORE_LineItem: CORE_LineItemQsum(i) = CORE_Q1Q1(join_CORE_LineItem_CORE_Q(i)) * 666};
s.t. C8 {i in CORE_S: CORE_Q1Q(i) == sum{ j in CORE_LineItem} objv_CORE_LineItemQ(j) * CORE_LineItemQ(j)};
s.t. C9 {i in CORE_S: 1 in CORE_LineItem: CORE_Q1Q(i) == objv_CORE_LineItemQ(i) * CORE_LineItemQ(i)};
s.t. C10 {i in CORE_LineItem: objv_CORE_LineItemQsum(i) == CORE_LineItemQsum(i)};
```

MathProg
or AMPL

Overview



Visualizations

Demo

```
RULES:
HChooseS(ok,pk,sk,qnt,sk',country) :- PartSupp(pk,sk') & LineItem(ok,pk,sk,qnt)
& SuppNation(sk2,country)
HLineItem(ok,pk,sk',qnt) :- HChooseS(ok,pk,sk,qnt,sk',country)
HOrderSum(country,count(*) ) :- HChooseS(ok,pk,sk,qnt,sk',country)
[c? <= 10] <- HOrderSum(country,c?)
MAXIMIZE(count(*) ) :- HChooseS(ok,pk,sk,qnt,sk,country)
```

TiQL

Language semantics

Evaluation of a TiQL program: Translation from TiQL to linear constraints

MathProg or AMPL

Performance optimizations

```
example10.mod
1 #
2 # Example problem
3 #
4 #
5 /* sets */
6 set CORE_LineItem;
7 set CORE_Q;
8 set LineItem;
9 set PartSupp;
10 set KEY_Q;
11 set KEY_Q_0;
12 set CORE_S;
13 set CORE_Obj;
14
15 /* parameters */
16 param join_CORE_LineItem2_CORE_Q {i in CORE_LineItem} :
17 param LineItemSquare {i in LineItem} :
18 param join_CORE_Q_LineItem3 {i in CORE_Q} :
19 param join_CORE_Q_PartSupp {i in CORE_Q} :
20 param key_CORE_Q_0 {i in KEY_Q_0, j in CORE_Q} :
21 param key_CORE_Q_1 {i in KEY_Q_1, j in CORE_Q} :
22 param obj_CORE_S_CORE_LineItem {i in CORE_S, j in CORE_LineItem} :
23
24 /* variables */
25 var CORE_Obj {i in CORE_Q} binary ;
26 var varKEY_Q_0 {i in KEY_Q_0} binary #1;
27 var varKEY_Q_1 {i in KEY_Q_1} binary #1;
28 var CORE_ObjSum {i in CORE_Q} :
29 var CORE_LineItem2 {i in CORE_LineItem} binary ;
30 var CORE_LineItemSquare {i in CORE_LineItem} :
31 var CORE_Obj {i in CORE_S} binary ;
32 var CORE_ObjSum {i in CORE_S} :
33 var objp_CORE_LineItemSum {i in CORE_LineItem} >= 0 ;
34 var CORE_ObjSum {i in CORE_Obj} binary ;
35 var CORE_ObjSum {i in CORE_Obj} :
36
37 /* constraints */
38 s.t. CORE_Q1 in KEY_Q_0: 1 >= sum{ j in CORE_Q} key_CORE_Q_0{1,j} * CORE_Obj{j};
39 s.t. CORE_Q1 in KEY_Q_1: 1 >= sum{ j in CORE_Q} key_CORE_Q_1{1,j} * CORE_Obj{j};
40 s.t. CORE_Q1 in KEY_Q_0: 1 <= sum{ j in CORE_Q} key_CORE_Q_0{1,j} * CORE_Obj{j};
41 s.t. CORE_Q1 in CORE_Q: CORE_ObjSum{1} = LineItemSquare{join_CORE_Q_LineItem3{1}};
42 s.t. CORE_Q1 in CORE_LineItem2: CORE_LineItemSum{1} = CORE_ObjSum{join_CORE_LineItem2_CORE_Q{1}};
43 s.t. CORE_Q1 in CORE_LineItem2: CORE_LineItemSquare{1} <= CORE_ObjSum{join_CORE_LineItem2_CORE_Q{1}} * 666;
44 s.t. CORE_Q1 in CORE_LineItem2: CORE_LineItemSum{1} <= CORE_ObjSum{join_CORE_LineItem2_CORE_Q{1}} * 666;
45 s.t. CORE_Q1 in CORE_LineItem2: CORE_LineItemSquare{1} >= CORE_ObjSum{join_CORE_LineItem2_CORE_Q{1}} * 666;
46 s.t. CORE_Q1 in CORE_S: CORE_ObjSum{1} <= sum{ j in CORE_LineItem2} objp_CORE_S_LineItem2{1,j} * CORE_LineItem2{1,j};
47 s.t. CORE_Q1 in CORE_S: j in CORE_LineItem2: CORE_ObjSum{1} >= objp_CORE_S_LineItem2{1,j} * CORE_LineItem2{1,j};
48 s.t. CORE_Q1 in CORE_LineItem2: objp_CORE_LineItemSum{1} <= CORE_LineItemSquare{1};
49 s.t. CORE_Q1 in CORE_LineItem2: objp_CORE_LineItemSum{1} <= CORE_LineItemSquare{1} * 666;
50 s.t. CORE_Q1 in CORE_LineItem2: objp_CORE_LineItemSum{1} >= CORE_LineItemSquare{1} * 666;
```

Tiresias Query Language

- Datalog-like notation:

$$\underbrace{P(\bar{x})}_{\text{head}} \text{ :- } \underbrace{B_1(\bar{x}_1) \wedge B_2(\bar{x}_2) \wedge \cdots \wedge B_n(\bar{x}_n)}_{\text{body: conjunction of predicates}}$$

- TiQL semantics:

Mapping from EDBs (Extensional Database) to possible worlds over **HDBs (Hypothetical Database)**

$$\text{HDB} \longrightarrow H P(\bar{x}) \text{ :- } body$$

TiQL Rules

Deduction Rule

$HP(\bar{x}) :- \text{body}$

Semantics:

Similar to repair-key semantics [Antonova et al. SIGMOD'07], [Koch ICDT'09]

Reduction Rule

$HP(\bar{x}) :< \text{body}$

Semantics:

Takes a subset of tuples

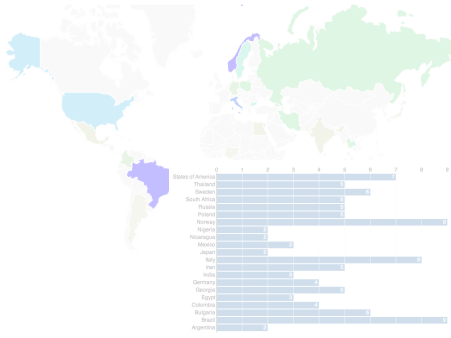
Constraint Rule

$[\text{arithm-pred}] <- \text{body}$

Semantics:

The head predicate needs to hold for all tuples

Overview



Visualizations

Demo

```

RULES:
HChooseS(ok,pk,sk,qnt,sk',country) :- PartSupp(pk,sk') & LineItem(ok,pk,sk,qnt)
& SuppNation(sk2,country)
HLineItem(ok,pk,sk',qnt) :- HChooseS(ok,pk,sk,qnt,sk',country)
HOrderSum(country,count(*) ) :- HChooseS(ok,pk,sk,qnt,sk',country)
[c? <= 10] <- HOrderSum(country,c?)
MAXIMIZE(count(*) ) :- HChooseS(ok,pk,sk,qnt,sk,country)
    
```

TiQL

Language semantics

Evaluation of a TiQL program: Translation from TiQL to linear constraints

MathProg or AMPL

Performance optimizations

```

example10.mod
1 #
2 # Example problem
3 #
4 #
5 /* sets */
6 set CORE_LineItem;
7 set CORE_Q;
8 set LineItem;
9 set PartSupp;
10 set KEY_Q;
11 set KEY_Q_0;
12 set CORE_S;
13 set CORE_Obj;
14
15 /* parameters */
16 param join_CORE_LineItem2_CORE_Q {i in CORE_LineItem} :
17   param LineItemSquare {i in LineItem} :
18   param join_CORE_Q_LineItem3 {i in CORE_Q} :
19   param join_CORE_Q_PartSupp {i in CORE_Q} :
20   param key_CORE_Q_0 {i in KEY_Q_0, j in CORE_Q} :
21   param key_CORE_Q_0 {i in KEY_Q_0, j in CORE_Q} :
22   param obj_CORE_S_CORE_LineItem {i in CORE_S, j in CORE_LineItem} :
23
24 /* variables */
25 var CORE_Obj {i in CORE_Q} binary ;
26 var varKEY_Q_0 {i in KEY_Q_0} binary #1;
27 var varKEY_Q_1 {i in KEY_Q_1} binary #1;
28 var CORE_ObjSum {i in CORE_Q} :
29 var CORE_LineItem2 {i in CORE_LineItem} binary ;
30 var CORE_LineItemSquare {i in CORE_LineItem} :
31 var CORE_Obj {i in CORE_S} binary ;
32 var CORE_ObjSum {i in CORE_S} :
33 var objp_CORE_LineItemSum {i in CORE_LineItem} >= 0 ;
34 var CORE_ObjSum {i in CORE_Obj} binary ;
35 var CORE_ObjSum {i in CORE_Obj} :
36
37 /* constraints */
38 s.t. CORE1 {i in KEY_Q_0: 1 >= sum{ j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Obj[j]};
39 s.t. CORE2 {i in KEY_Q_1: 1 >= sum{ j in CORE_Q} key_CORE_Q_1[i,j] * CORE_Obj[j]};
40 s.t. CORE3 {i in KEY_Q_0: 1 <= sum{ j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Obj[j]};
41 s.t. CORE4 {i in CORE_Q: CORE_ObjSum[i] = LineItemSquare[join_CORE_Q_LineItem2(i)]};
42 s.t. CORE5 {i in CORE_LineItem: CORE_LineItem2[i] = CORE_Obj[join_CORE_LineItem2_CORE_Q(i)]};
43 s.t. CORE6 {i in CORE_LineItem: CORE_LineItemSquare[i] <= CORE_ObjSum[join_CORE_LineItem2_CORE_Q(i)]};
44 s.t. CORE7 {i in CORE_LineItem: CORE_LineItemSquare[i] <= CORE_ObjSum[join_CORE_LineItem2_CORE_Q(i)]} * 666;
45 s.t. CORE8 {i in CORE_LineItem: CORE_LineItemSquare[i] <= CORE_ObjSum[join_CORE_LineItem2_CORE_Q(i)]} * 666;
46 s.t. CORE9 {i in CORE_LineItem: CORE_LineItemSquare[i] <= CORE_ObjSum[join_CORE_LineItem2_CORE_Q(i)]} * 666;
47 s.t. CORE10 {i in CORE_S: CORE_ObjSum[i] <= sum{ j in CORE_LineItem} objp_CORE_S_LineItem2[i,j] * CORE_LineItem2[j]};
48 s.t. CORE11 {i in CORE_S: j in CORE_LineItem: CORE_ObjSum[i] <= objp_CORE_S_LineItem2[i,j] * CORE_LineItem2[j]};
49 s.t. CORE12 {i in CORE_LineItem: objp_CORE_LineItemSum[i] <= CORE_LineItemSquare[i]};
50 s.t. CORE13 {i in CORE_LineItem: objp_CORE_LineItemSum[i] <= CORE_LineItemSquare[i]};
51
    
```

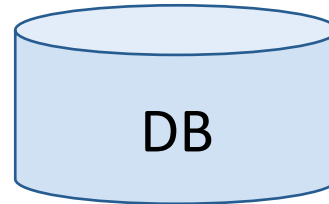
Evaluating a TiQL Program

TiQL

RULES:

```
HChooseS(ok,pk,sk,qnt,sk',country) :- PartSupp(pk,sk') & LineItem(ok,pk,sk,qnt)
                                     & SuppNation(sk2,country)

HLineItem(ok,pk,sk',qnt)             :- HChooseS(ok,pk,sk,qnt,sk',country)
HOrderSum(country,count(*))          :- HChooseS(ok,pk,sk,qnt,sk',country)
[c? <= 10]                            <- HOrderSum(country,c?)
MAXIMIZE(count(*)) :- HChooseS(ok,pk,sk,qnt,sk, country)
```

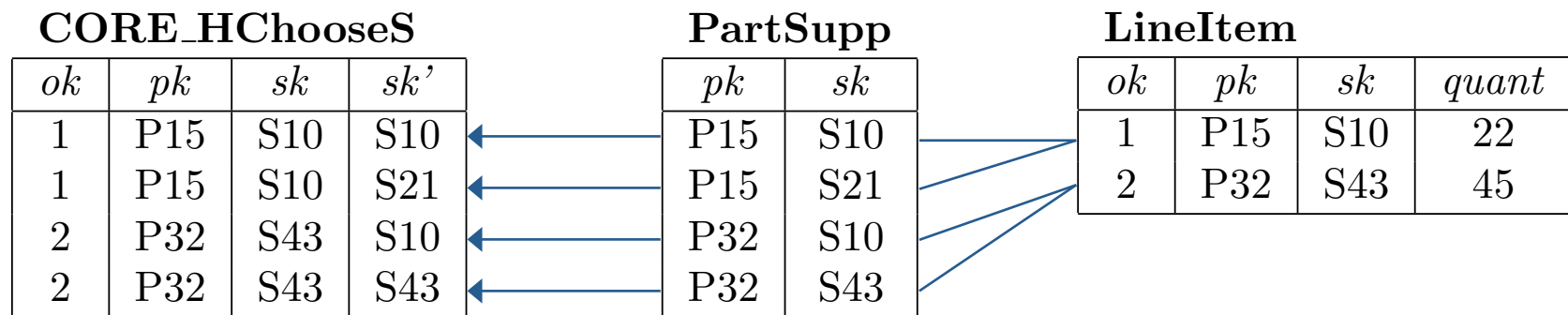


Mixed Integer Program
(MIP)

```
1 #
2 # Example problem
3 #
4 #
5 /* sets */
6 set CORE_LineItem2;
7 set CORE_Q;
8 set LineItem2;
9 set PartSupp;
10 set KEY_Q;
11 set KEY_Q_1;
12 set CORE_S;
13 set CORE_0;
14
15 /* parameters */
16 param join_CORE_LineItem2_CORE_Q {i in CORE_LineItem2} :-
17   param LineItemQuant {i in LineItem2} :-
18   param join_CORE_Q_LineItem2 {i in CORE_Q} :-
19   param join_CORE_Q_PartSupp {i in CORE_Q} :-
20   param key_CORE_Q_1 {i in KEY_Q_1, j in CORE_Q} :-
21   param key_CORE_Q_1 {i in KEY_Q_1, j in CORE_Q} :-
22   param opt_CORE_S_CORE_LineItem2 {i in CORE_S, j in CORE_LineItem2} :-
23
24 /* variables */
25 var CORE_01id {i in CORE_Q} binary :-
26   var varKEY_Q_1 {i in KEY_Q_1} binary :-
27   var varKEY_Q_1 {i in KEY_Q_1} binary :-
28   var CORE_0quant {i in CORE_Q} :-
29   var CORE_LineItem2id {i in CORE_LineItem2} binary :-
30   var CORE_LineItemQuant {i in CORE_LineItem2} :-
31   var CORE_01id {i in CORE_S} :-
32   var pppp_CORE_LineItemQuant {i in CORE_LineItem2} >= 0 :-
33   var CORE_01id {i in CORE_0} binary :-
34   var CORE_01id {i in CORE_0} :-
35
36 /* constraints */
37 s.t. c228 {i in KEY_Q_1} 1 >= sum{j in CORE_Q} key_CORE_Q_1[i,j] * CORE_01id[j];
38 s.t. c229 {i in KEY_Q_1} 1 >= sum{j in CORE_Q} key_CORE_Q_1[i,j] * CORE_01id[j];
39 s.t. c230 {i in KEY_Q_1} 1 >= sum{j in CORE_Q} key_CORE_Q_1[i,j] * CORE_01id[j];
40 s.t. c231 {i in CORE_Q} CORE_0quant[i] = LineItemQuant[join_CORE_LineItem2(i)];
41 s.t. c232 {i in CORE_Q} CORE_LineItem2id[i] = CORE_LineItem2id[join_CORE_LineItem2(i)];
42 s.t. c233 {i in CORE_LineItem2} CORE_LineItemQuant[i] = CORE_LineItemQuant[join_CORE_LineItem2(i)];
43 s.t. c234 {i in CORE_LineItem2} CORE_LineItemQuant[i] = CORE_LineItemQuant[join_CORE_LineItem2(i)];
44 s.t. c235 {i in CORE_LineItem2} CORE_LineItemQuant[i] = CORE_LineItemQuant[join_CORE_LineItem2(i)] * 666;
45 s.t. c236 {i in CORE_LineItem2} CORE_LineItemQuant[i] = CORE_LineItemQuant[join_CORE_LineItem2(i)] * 666;
46 s.t. c237 {i in CORE_S} CORE_01id[i] = sum{j in CORE_LineItem2} opt_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2id[j];
47 s.t. c238 {i in CORE_S} 1 >= sum{j in CORE_LineItem2} opt_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2id[j];
48 s.t. c239 {i in CORE_LineItem2} pppp_CORE_LineItemQuant[i] = CORE_LineItemQuant[i];
49 s.t. c240 {i in CORE_LineItem2} CORE_LineItemQuant[i] = CORE_LineItemQuant[i];
50 s.t. c241 {i in CORE_LineItem2} CORE_LineItemQuant[i] = CORE_LineItemQuant[i];
```

Evaluating a TiQL Program

$HChooseS(ok, pk, sk, sk') :- PartSupp(pk, sk') \ \& \ LineItem(ok, pk, sk, qnt)$



possible worlds

<i>ok</i>	<i>pk</i>	<i>sk</i>	<i>sk'</i>
1	P15	S10	S10
2	P32	S43	S43

<i>ok</i>	<i>pk</i>	<i>sk</i>	<i>sk'</i>
1	P15	S10	S21
2	P32	S43	S43

<i>ok</i>	<i>pk</i>	<i>sk</i>	<i>sk'</i>
1	P15	S10	S10
2	P32	S43	S10

<i>ok</i>	<i>pk</i>	<i>sk</i>	<i>sk'</i>
1	P15	S10	S21
2	P32	S43	S10

Key Constraints

CORE_HChooseS

ok	pk	sk	sk'	
1	P15	S10	S10	x_1
1	P15	S10	S21	x_2
2	P32	S43	S10	x_3
2	P32	S43	S43	x_4

NOT a possible world

ok	pk	sk	sk'
1	P15	S10	S10
1	P15	S10	S21

$Key(ok, pk, sk)$

$$x_1 + x_2 \leq 1$$

$$x_3 + x_4 \leq 1$$

$$\forall k_j : \quad 0 \leq x_i \leq 1$$

$$\sum_{i, key(x_i)=k_j} x_i \leq 1$$

Provenance Constraints

- A TiQL rule specifies transformations
- Transformations define provenance
 - ⊙ Boolean semantics for queries without aggregates
 - ⊙ Semi-module provenance for queries with aggregates [Amsterdamer et al. PODS'11]

Disjunction:

$$Y = X_1 \vee X_2 \vee \dots \vee X_n$$

$$\forall i, y \geq x_i$$

$$y \leq \sum_i x_i$$

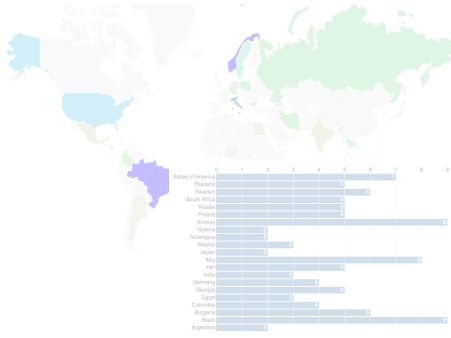
Conjunction:

$$Y = X_1 \wedge X_2 \wedge \dots \wedge X_n$$

$$\forall i, y \leq x_i$$

$$y \geq \sum_i x_i - (n - 1)$$

Overview



Visualizations

Demo

```
RULES:
HChooseS(ok,pk,sk,qnt,sk',country) :- PartSupp(pk,sk') & LineItem(ok,pk,sk,qnt)
& SuppNation(sk2,country)
HLineItem(ok,pk,sk',qnt) :- HChooseS(ok,pk,sk,qnt,sk',country)
HOrderSum(country,count(*) ) :- HChooseS(ok,pk,sk,qnt,sk',country)
[c? <= 10] <- HOrderSum(country,c?)
MAXIMIZE(count(*) ) :- HChooseS(ok,pk,sk,qnt,sk,country)
```

TiQL

Language semantics

Evaluation of a TiQL program: Translation from TiQL to linear constraints

MathProg or AMPL

Performance optimizations

```
example10.mod
1 #
2 # Example problem
3 #
4
5 /* sets */
6 set CORE_LineItem;
7 set CORE_Q;
8 set LineItem;
9 set PartSupp;
10 set KEY_Q;
11 set KEY_Q_0;
12 set CORE_S;
13 set CORE_OB;
14
15 /* parameters */
16 param join_CORE_LineItem2_CORE_Q {i in CORE_LineItem} :
17 param LineItemSquare {i in LineItem} :
18 param join_CORE_Q_LineItem3 {i in CORE_Q} :
19 param join_CORE_Q_PartSupp {i in CORE_Q} :
20 param key_CORE_Q_0 {i in KEY_Q_0, j in CORE_Q} :
21 param key_CORE_Q_1 {i in KEY_Q_1, j in CORE_Q} :
22 param join_CORE_S_CORE_LineItem {i in CORE_S, j in CORE_LineItem} :
23
24 /* variables */
25 var CORE_Qid {i in CORE_Q} binary ;
26 var vKEY_Q_0 {i in KEY_Q_0} binary #1;
27 var vKEY_Q_1 {i in KEY_Q_1} binary #1;
28 var CORE_Qquant {i in CORE_Q} :
29 var CORE_LineItem2id {i in CORE_LineItem} binary ;
30 var CORE_LineItemSquare {i in CORE_LineItem} :
31 var CORE_Qid {i in CORE_S} binary ;
32 var CORE_Sx {i in CORE_S} :
33 var objv_CORE_LineItemQuant {i in CORE_LineItem} >= 0 ;
34 var CORE_Qid {i in CORE_OB} binary ;
35 var CORE_OB {i in CORE_OB} :
36
37 /* constraints */
38 s.t. CORE_Qid in KEY_Q_0: 1 >= sum{ i in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qid[i];
39 s.t. CORE_Qid in KEY_Q_1: 1 >= sum{ i in CORE_Q} key_CORE_Q_1[i,j] * CORE_Qid[i];
40 s.t. CORE_Qid in CORE_Q: 1 <= sum{ i in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qid[i];
41 s.t. CORE_Qid in CORE_Q: CORE_Qquant[i] = LineItemSquare[join_CORE_Q_LineItem2(i)];
42 s.t. CORE_Qid in CORE_LineItem2: CORE_LineItem2id[i] = CORE_Qid[join_CORE_LineItem2_CORE_Q(i)];
43 s.t. CORE_Qid in CORE_LineItem2: CORE_LineItemSquare[i] <= CORE_Qquant[join_CORE_LineItem2_CORE_Q(i)] * 666;
44 s.t. CORE_Qid in CORE_LineItem2: CORE_LineItemSquare[i] <= CORE_Qid[join_CORE_LineItem2_CORE_Q(i)] * 666;
45 s.t. CORE_Qid in CORE_LineItem2: CORE_LineItemSquare[i] <= CORE_Qquant[join_CORE_LineItem2_CORE_Q(i)] * 666;
46 s.t. CORE_Qid in CORE_LineItem2: CORE_LineItemSquare[i] <= CORE_Qid[join_CORE_LineItem2_CORE_Q(i)] * 666;
47 s.t. CORE_Qid in CORE_S: 1 >= sum{ i in CORE_LineItem2} objv_CORE_S_LineItem2[i] * CORE_LineItem2id[i];
48 s.t. CORE_Qid in CORE_S: 1 >= sum{ i in CORE_LineItem2} CORE_Qid[i] <= objv_CORE_S_LineItem2[i] * CORE_LineItem2id[i];
49 s.t. CORE_Qid in CORE_LineItem2: objv_CORE_LineItemQuant[i] <= CORE_LineItemSquare[i];
50 s.t. CORE_Qid in CORE_LineItem2: objv_CORE_LineItemQuant[i] <= CORE_LineItemSquare[i] * 666;
```

Optimizing Performance

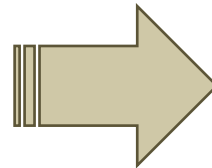
○ Model optimizer

- ⊙ eliminates variables, constraints, and parameters
- ⊙ uses **key constraints**, **functional dependencies**, and **provenance**

Significantly faster than letting the MIP solver do it

○ Partitioning optimizer

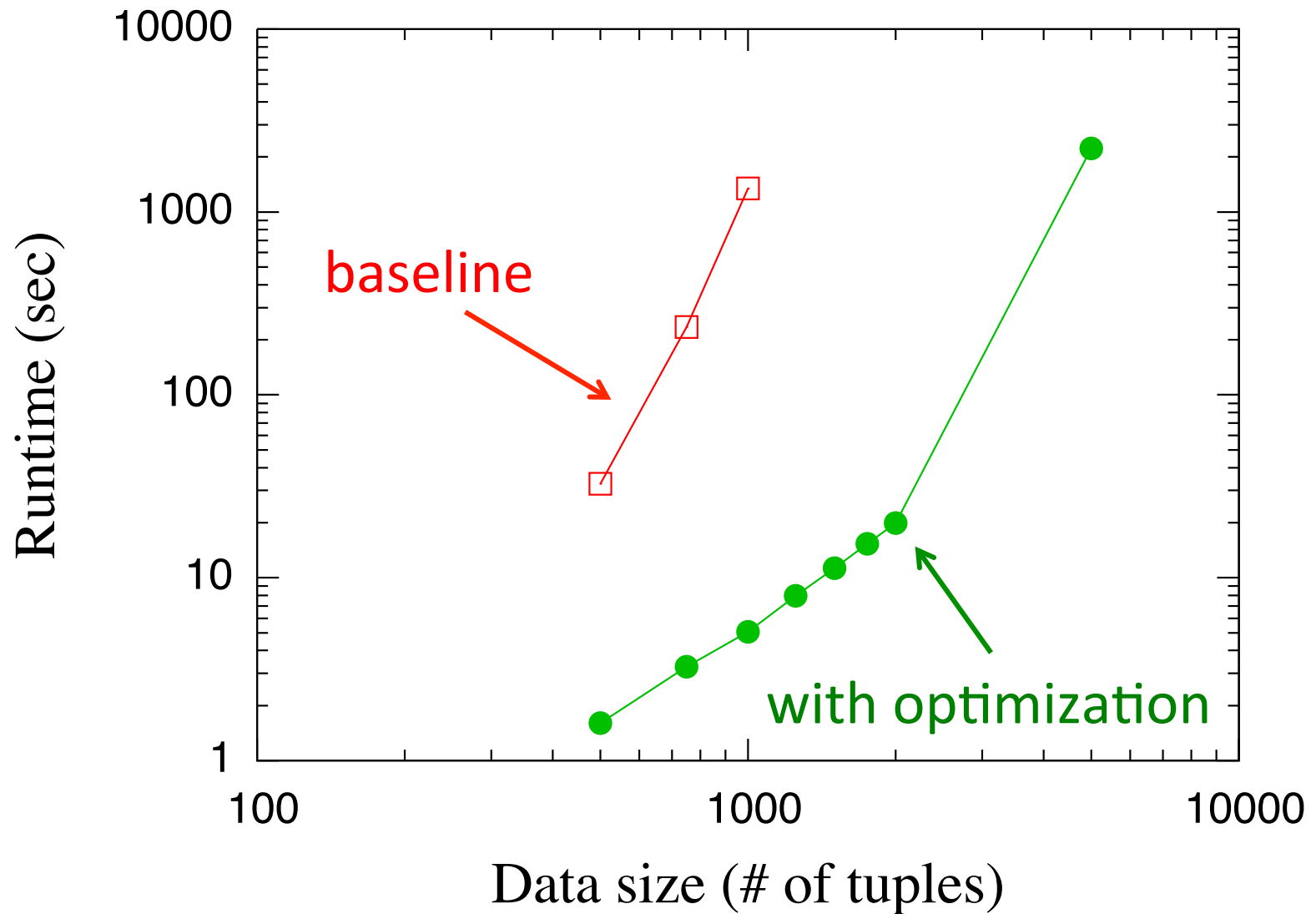
$$\begin{array}{l} \max(x_1 + x_2 + x_3 + x_4) \\ s.t : \\ \quad x_1 + x_2 \leq 50 \\ \quad x_3 + x_4 \leq 50 \\ \quad x_i \geq 0 \end{array}$$



$$\begin{array}{l} \max(x_1 + x_2) \\ s.t : \\ \quad x_1 + x_2 \leq 50 \\ \quad x_i \geq 0 \end{array}$$

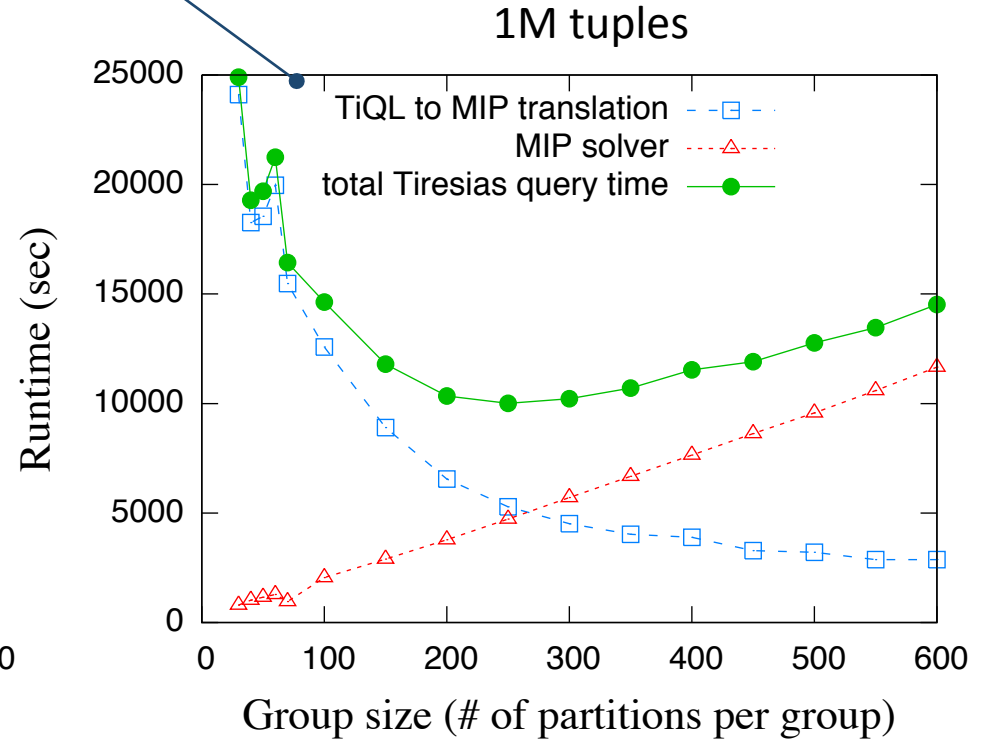
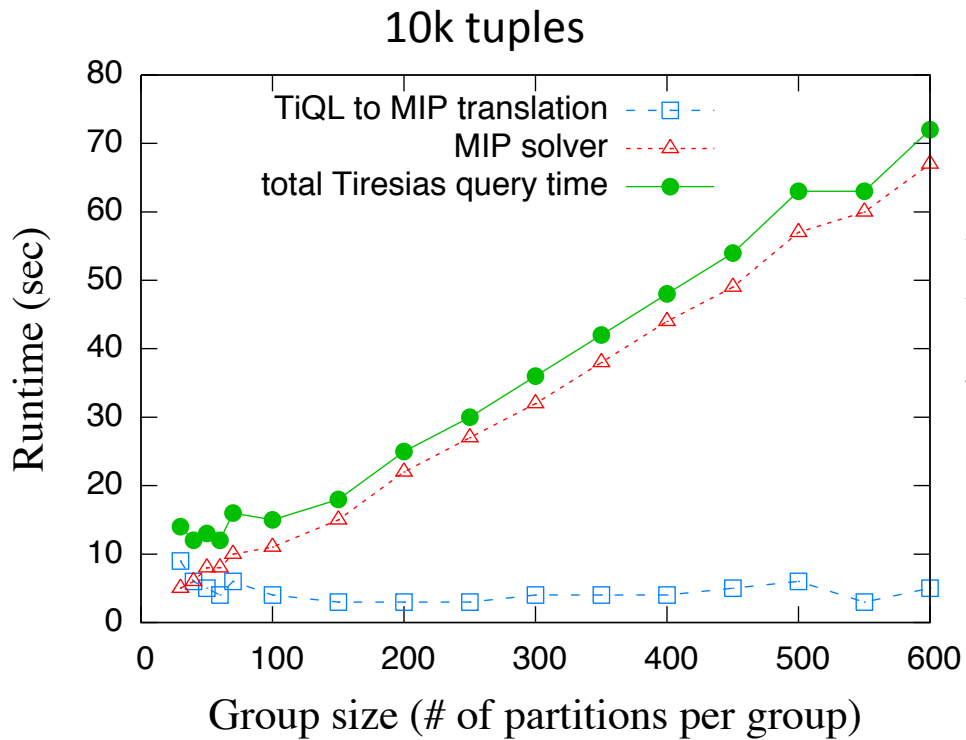
$$\begin{array}{l} \max(x_3 + x_4) \\ s.t : \\ \quad x_3 + x_4 \leq 50 \\ \quad x_i \geq 0 \end{array}$$

Evaluation of the Model Optimizer



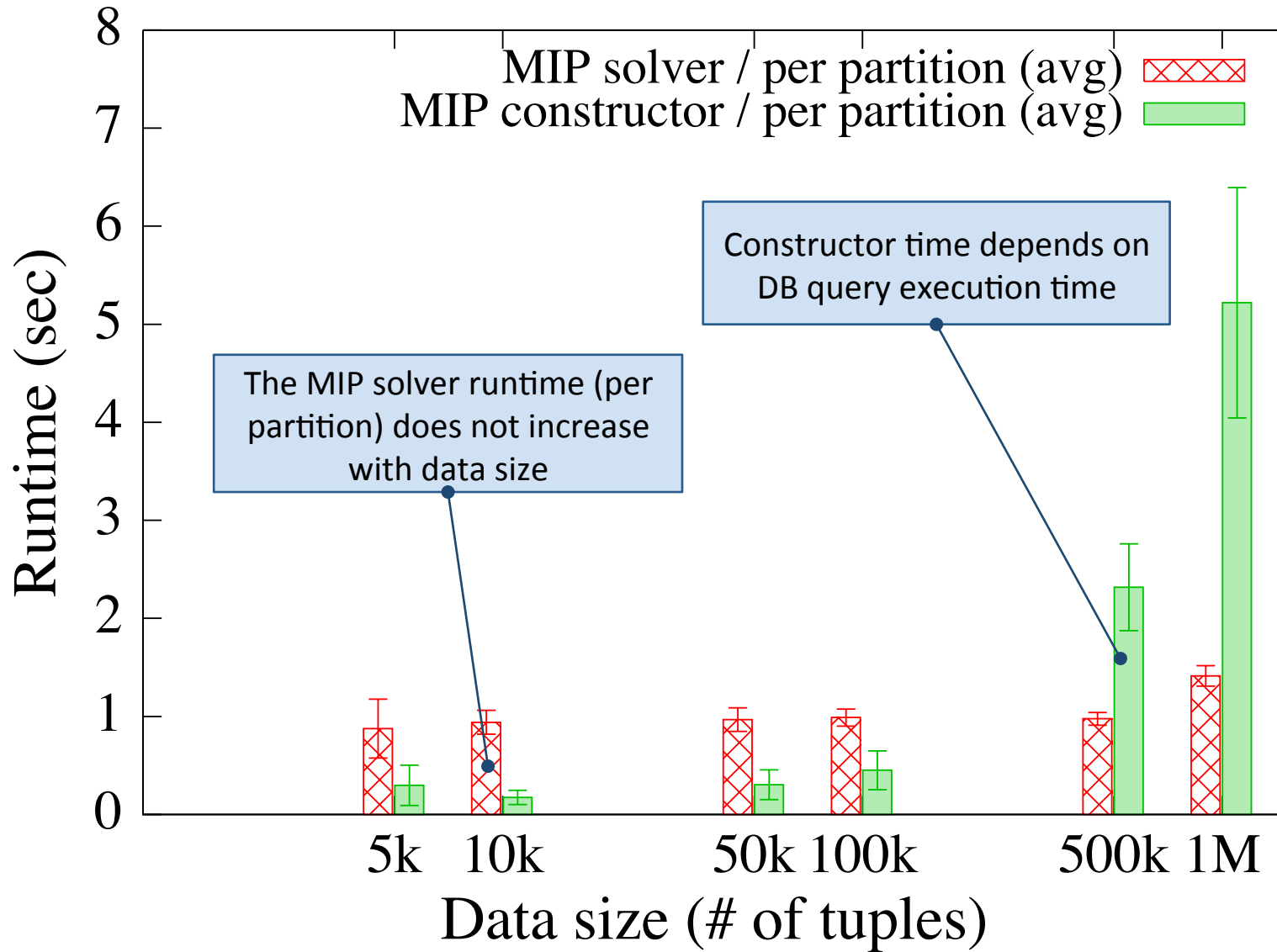
Evaluation of Tiresias Partitioning

complex dependency on the granularity of partitioning



granularity of partitioning

Scalability



Related Work

- Provenance

[Amsterdamer et al. PODS'11], [Cui et al. TODS'00],
[Green et al. PODS'07]

- Incomplete databases

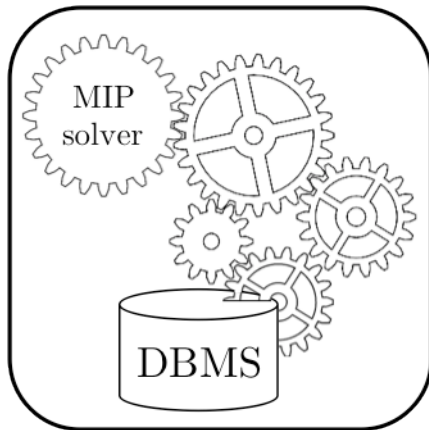
[Antonova et al. SIGMOD'07], [Imielinski et al. JACM'84],
[Koch ICDT'09]

- Other RDM problems

[Arasu et al. SIGMOD'11], [Binnig et al. ICDE'07],
[Bohannon et al. PODS'06], [Fagin et al. JACM'10]

Next Steps with Tiresias

Tiresias



Handling non-partitionable problems

Approximations

Parallelization and handling of skew

Result analysis and feedback-based
problem generation

SIGMOD Demo Group C

Location: Vaquero A
Time: 13:30-15:00

The image shows a screenshot of the Tiresias web application interface. The main window displays the 'Tiresias: How-To Queries' page, which includes a 'Predefined How-To Queries' section and a 'List of orders (Linitem)' table. A 'submit' button is visible. Below the table, there is a world map visualization showing the distribution of orders by country. A smaller window in the foreground shows the 'Tiresias: Admin View' page, which displays the 'Last Execution' results, including the execution time (593 msec), the displayed aggregate query, and relevant files. The 'TiQL' section shows the query used for the execution.

Tiresias: How-To Queries

How-To Queries Admin Panel

Predefined How-To Queries

1: Reduce quantities. Total quantity per order <= 150 items

submit

List of orders (Linitem)

orderkey	partkey	supplierkey	quantity
1	800 Plus		
1	2490 Power Series plus		
1	STARDOM VDS		
1	Esarge		
1	ProSale-RS		
1	Realtime Production Organizer		
1	Exaltas		
1	Edgewise Panel/Meters		
1	Exaploit		

Tiresias: Admin View

How-To Queries Admin Panel

Last Execution

Time: 593 msec

Displayed Aggregate View More

```
SELECT b.country AS name, count(*) AS value,
<version> AS version FROM <newTableName> a,
suppnation b WHERE a.version = <version> and
a.supplierkey=b.sk GROUP BY country;
```

Relevant Files

- MathProg_model
- Solution file
- Solver log
- CORE SQL

TiQL View More

```
Q(ok, pk, sk, quant, sk2, country) :- LinItem0(ok, pk, sk, quant) &
PartSupp10(pk, sk2) & SuppNation(sk2, country)

LinItem2(ok, pk, sk2, quant) :- Q(ok, pk, sk, quant, sk2, country)

S(country, count(*)) :- Q(ok, pk, sk, quant, sk2, country)

c <= 7 :- S(country, c)

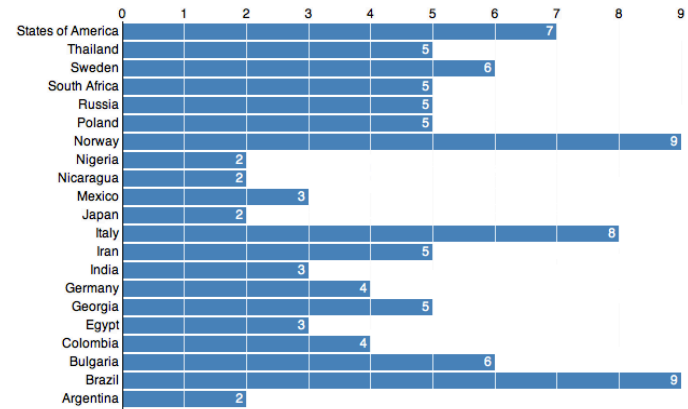
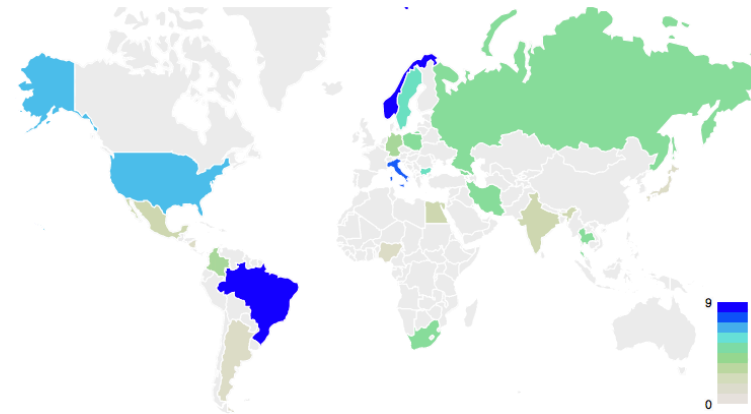
Obj(count(*)) :- Q(ok, pk, sk, quant, sk, country)

maximize Obj
```

Tiresias - SIGMOD 2012 <http://db.cs.washington.edu/tiresias/>

Contributions

- How-To queries
- Using MIP solvers to answer How-To queries
- Tiresias prototype implementation



<http://db.cs.washington.edu/tiresias>