Tiresias

The Database Oracle for How-To Queries

Alexandra Meliou[§❖]

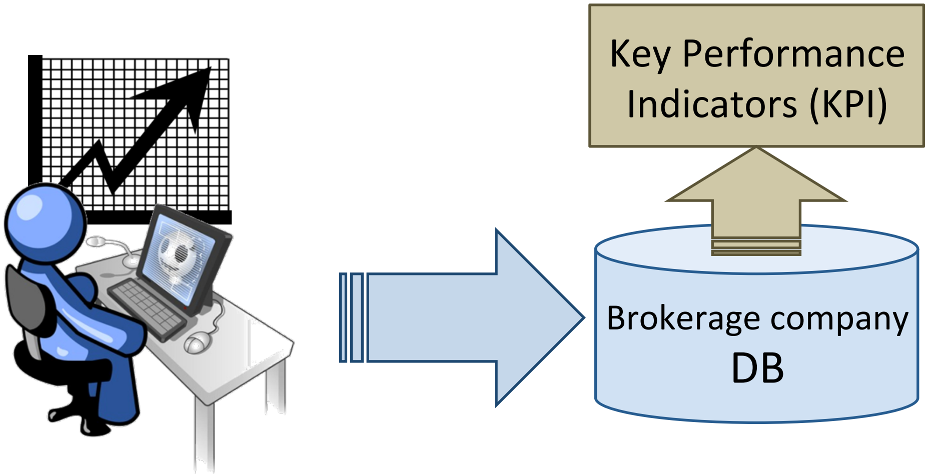
Dan Suciu[❖]

[§]*University of Massachusetts Amherst*

[❖]*University of Washington*

University of Washington
Database Group

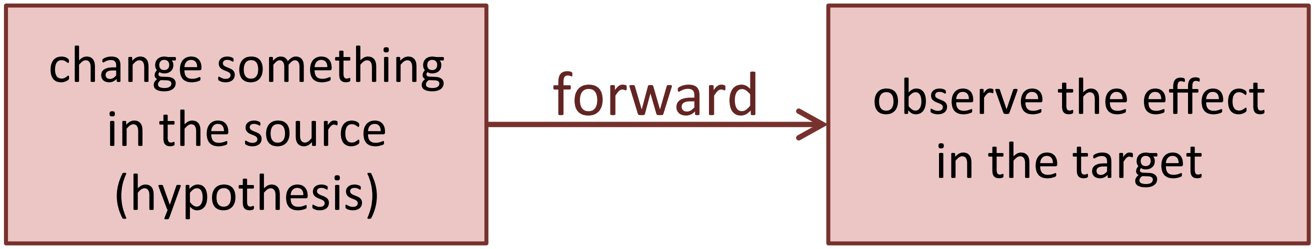
Hypothetical (What-if) Queries



Key Performance Indicators (KPI)

Brokerage company DB

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.”

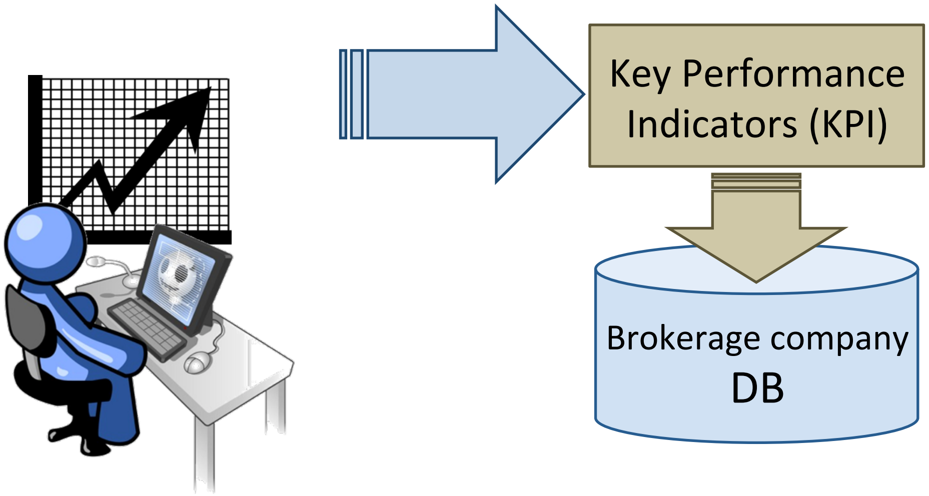


change something in the source (hypothesis)

forward

observe the effect in the target

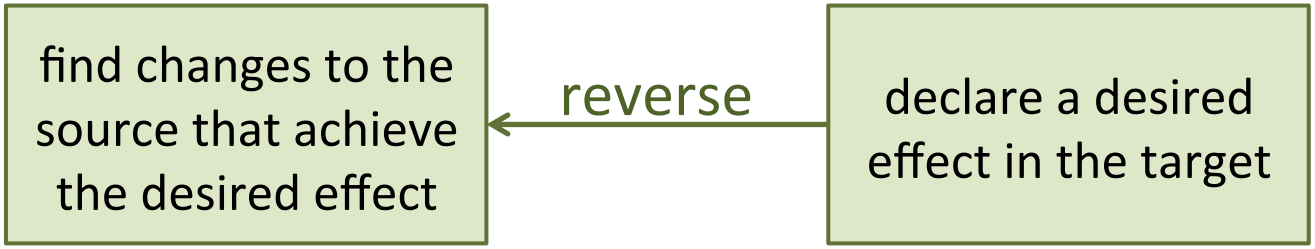
How-To Queries



Key Performance Indicators (KPI)

Brokerage company DB

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



find changes to the source that achieve the desired effect

reverse

declare a desired effect in the target

TPC-H example

- A manufacturing company keeps records of inventory orders in a LineItem 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)

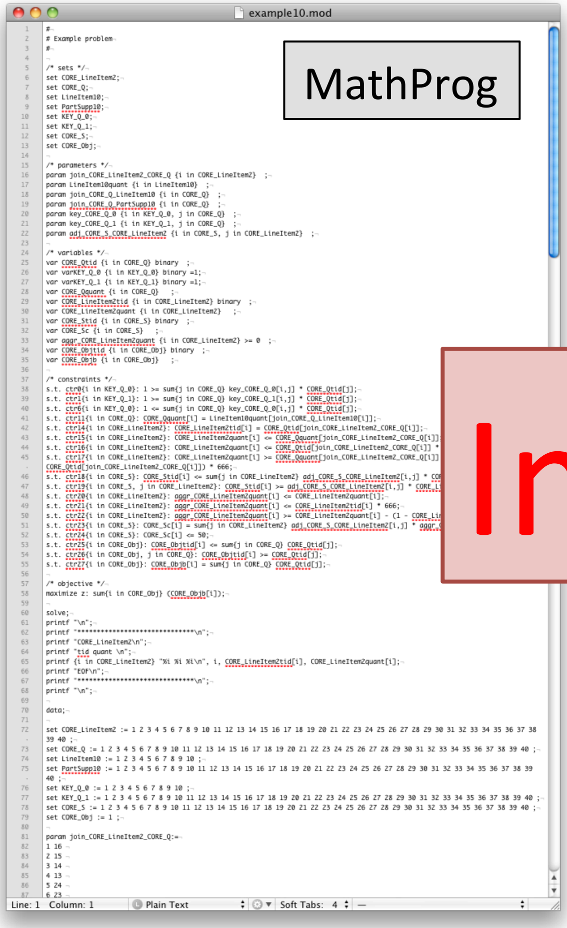
  - Minimize the number of changes

(optimization objective)

constraint optimization

Constraint Optimization on Big Data

this is for a set of
10 lineitems and 40
suppliers

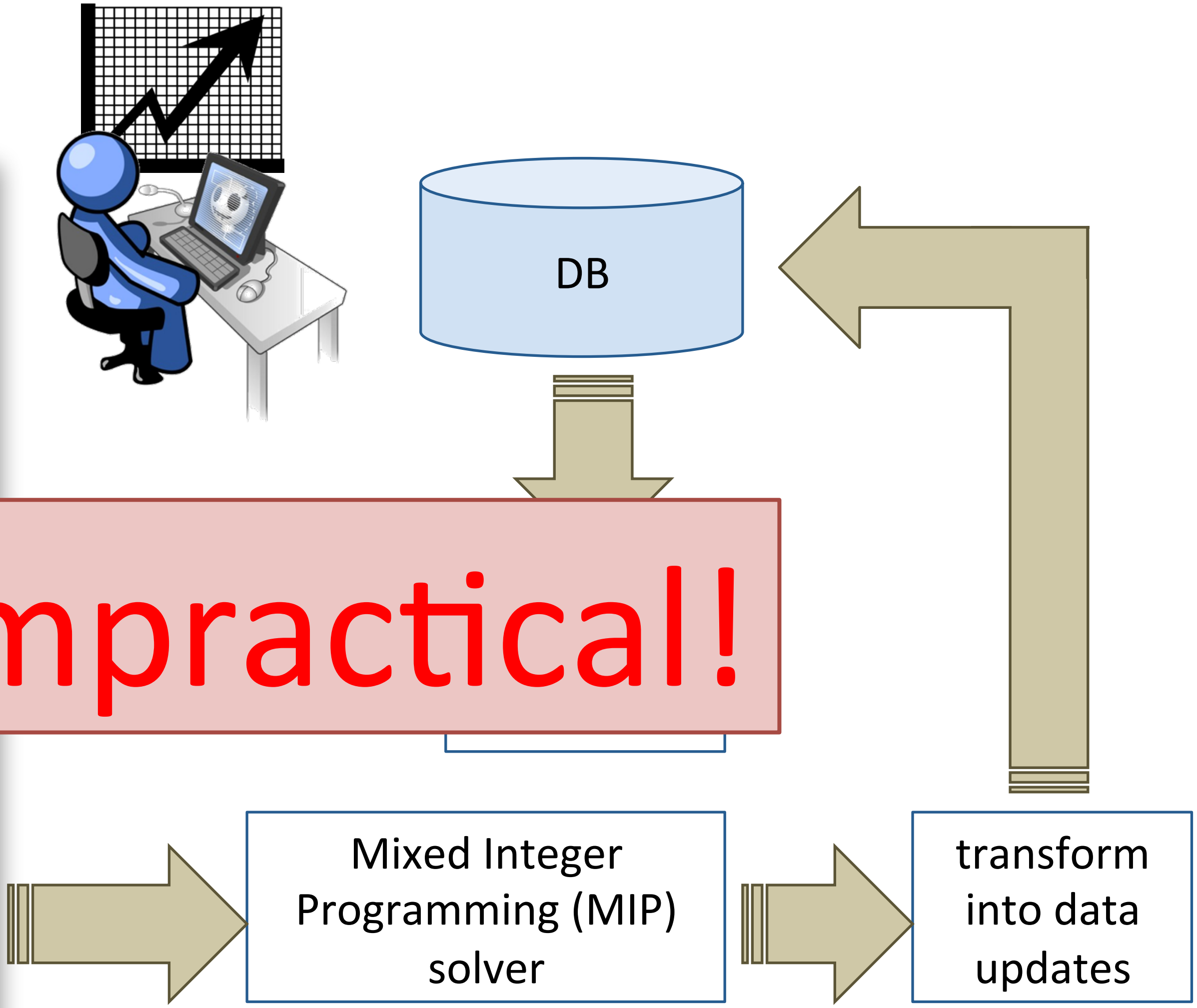


MathProg

example10.mod

```
#
# Example problem
#

/* sets */
set CORE_LineItem2;
set CORE_Q;
set LineItem10;
set PartSupp10;
set KEY_Q_0;
set KEY_Q_1;
set CORE_S;
set CORE_Obj;

/* parameters */
param join_CORE_LineItem2_CORE_Q {i in CORE_LineItem2}  ;
param LineItem10quant {i in LineItem10}  ;
param join_CORE_Q_LineItem10 {i in CORE_Q}  ;
param join_CORE_Q_PartSupp10 {i in CORE_Q}  ;
param key_CORE_Q_0 {i in KEY_Q_0, j in CORE_Q}  ;
param key_CORE_Q_1 {i in KEY_Q_1, j in CORE_Q}  ;
param adj_CORE_S_CORE_LineItem2 {i in CORE_S, j in CORE_LineItem2}  ;

/* variables */
var CORE_Qtid {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_LineItem2tid {i in CORE_LineItem2} binary  ;
var CORE_LineItem2quant {i in CORE_LineItem2}   ;
var CORE_Stid {i in CORE_S} binary  ;
var CORE_Sc {i in CORE_S}   ;
var aggr_CORE_LineItem2quant {i in CORE_LineItem2} >= 0  ;
var CORE_Objtid {i in CORE_Obj} binary  ;
var CORE_Objb {i in CORE_Obj}   ;

/* constraints */
s.t. ctr0{i in KEY_Q_0}: 1 >= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr1{i in KEY_Q_1}: 1 >= sum{j in CORE_Q} key_CORE_Q_1[i,j] * CORE_Qtid[j];
s.t. ctr6{i in KEY_Q_0}: 1 <= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr11{i in CORE_Q}: CORE_Qquant[i] = LineItem10quant[join_CORE_Q_LineItem10[i]];
s.t. ctr14{i in CORE_LineItem2}: CORE_LineItem2tid[i] = CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]];
s.t. ctr15{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]]
s.t. ctr16{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]] *
s.t. ctr17{i in CORE_LineItem2}: CORE_LineItem2quant[i] >= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]]
CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]]) * 666;
s.t. ctr18{i in CORE_S}: CORE_Stid[i] <= sum{j in CORE_LineItem2} adj_CORE_S_CORE_LineItem2[i,j] * COR
s.t. ctr19{i in CORE_S, j in CORE_LineItem2}: CORE_Stid[i] >= adj_CORE_S_CORE_LineItem2[i,j] * CORE_Li
s.t. ctr20{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2quant[i];
s.t. ctr21{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2tid[i] * 666;
s.t. ctr22{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] >= CORE_LineItem2quant[i] - (1 - CORE_Li
s.t. ctr23{i in CORE_S}: CORE_Sc[i] = sum{j in CORE_LineItem2} adj_CORE_S_CORE_LineItem2[i,j] * aggr_C
s.t. ctr24{i in CORE_S}: CORE_Sc[i] <= 50;
s.t. ctr25{i in CORE_Obj}: CORE_Objtid[i] <= sum{j in CORE_Q} CORE_Qtid[j];
s.t. ctr26{i in CORE_Obj, j in CORE_Q}: CORE_Objtid[i] >= CORE_Qtid[j];
s.t. ctr27{i in CORE_Obj}: CORE_Objb[i] = sum{j in CORE_Q} CORE_Qtid[j];

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

solve;
printf "\n";
printf "******************************\n";
printf "CORE_LineItem2\n";
printf "tid quant \n";
printf {i in CORE_LineItem2} "%i %i %i\n", i, CORE_LineItem2tid[i], CORE_LineItem2quant[i];
printf "EOF\n";
printf "******************************\n";
printf "\n";

data;

set CORE_LineItem2 := 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 LineItem10 := 1 2 3 4 5 6 7 8 9 10 ;
set PartSupp10 := 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_Obj := 1 ;

param join_CORE_LineItem2_CORE_Q:=
1 16
2 15
3 14
4 13
5 24
6 23
```

Line: 1 Column: 1
Plain Text
Soft Tabs: 4



DB
Mixed Integer Programming (MIP) solver
transform into data updates

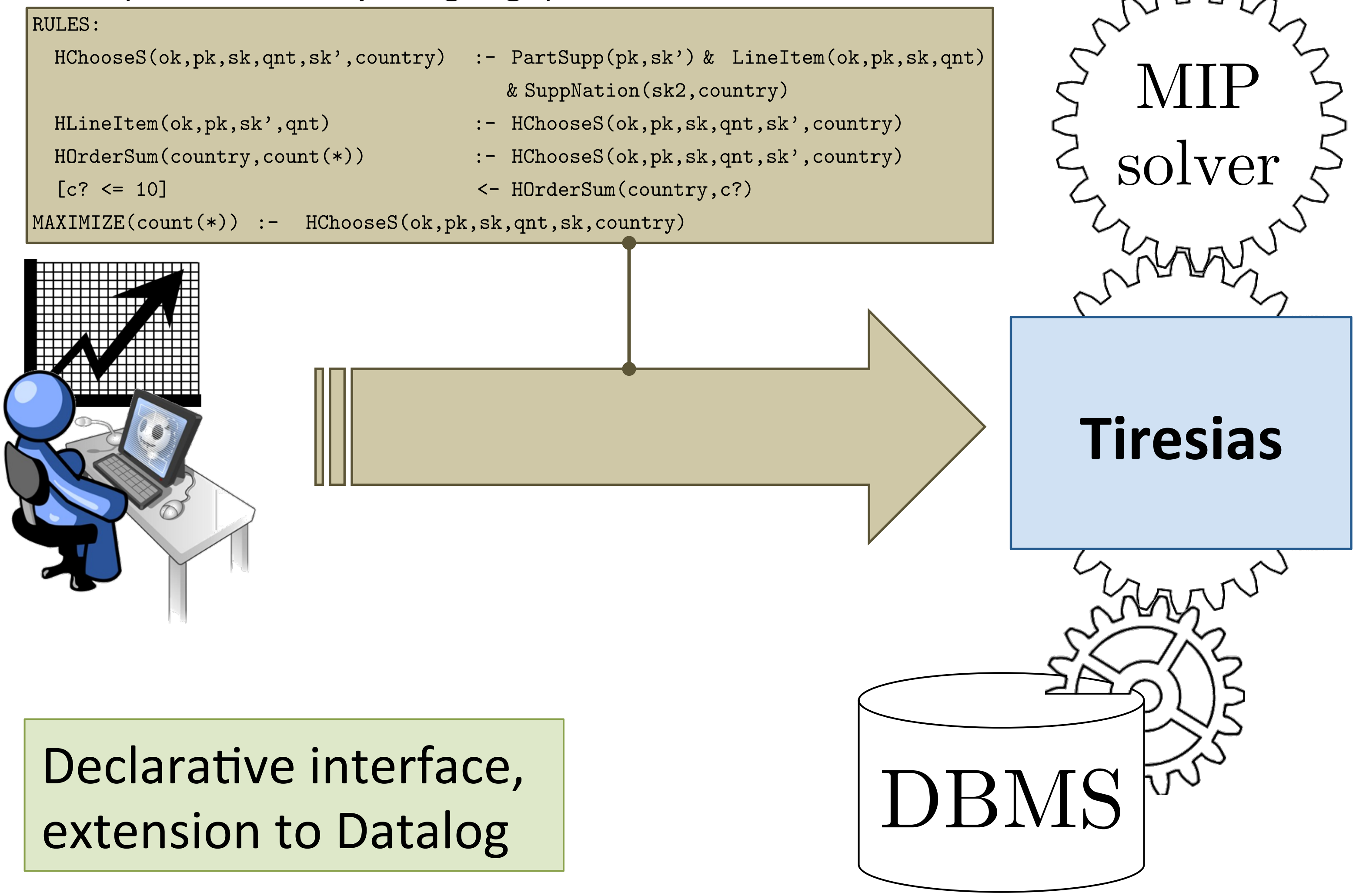
Impractical!

Demo: **Tiresias**

a tool that makes
how-to queries practical

Tiresias: How-To Query Engine

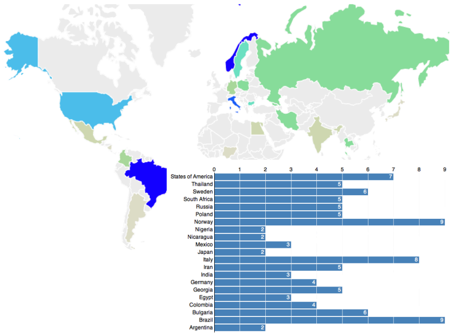
TiQL (Tiresias Query Language)



```
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

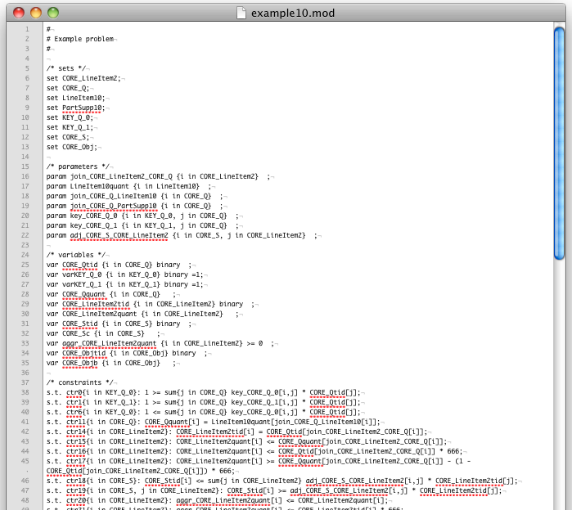
Overview



Visualizations

```
RULES:
  HChooseS(ok,pk,sk,qnt,sk',country)   :- PartSupp(pk,sk') & 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)
```

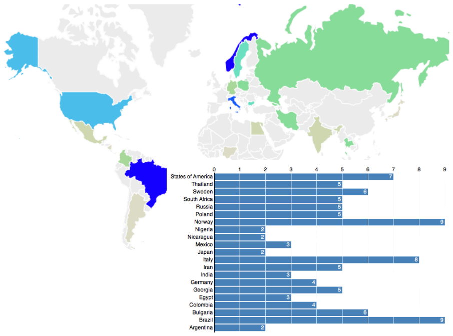
TiQL



```
#
# Example problem
#

/* sets */
set CORE_LineItem2;
set CORE_Q;
set LineItem10;
set PartSupp10;
set KEY_Q_0;
set KEY_Q_1;
set CORE_S;
set CORE_Obj;

/* parameters */
param join_CORE_LineItem2_CORE_Q {i in CORE_LineItem2}  ;
param LineItem10quant {i in LineItem10}  ;
param join_CORE_Q_LineItem10 {i in CORE_Q}  ;
param join_CORE_Q_PartSupp10 {i in CORE_Q}  ;
param key_CORE_Q_0 {i in KEY_Q_0, j in CORE_Q}  ;
param key_CORE_Q_1 {i in KEY_Q_1, j in CORE_Q}  ;
param adj_CORE_S_CORE_LineItem2 {i in CORE_S, j in CORE_LineItem2}  ;

/* variables */
var CORE_Qtid {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_LineItem2tid {i in CORE_LineItem2} binary  ;
var CORE_LineItem2quant {i in CORE_LineItem2}   ;
var CORE_Stid {i in CORE_S} binary  ;
var CORE_Sc {i in CORE_S}   ;
var aggr_CORE_LineItem2quant {i in CORE_LineItem2} >= 0  ;
var CORE_Objtid {i in CORE_Obj} binary  ;
var CORE_Objb {i in CORE_Obj}   ;

/* constraints */
s.t. ctr0{i in KEY_Q_0}: 1 >= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr1{i in KEY_Q_1}: 1 >= sum{j in CORE_Q} key_CORE_Q_1[i,j] * CORE_Qtid[j];
s.t. ctr6{i in KEY_Q_0}: 1 <= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr11{i in CORE_Q}: CORE_Qquant[i] = LineItem10quant[join_CORE_Q_LineItem10[i]];
s.t. ctr14{i in CORE_LineItem2}: CORE_LineItem2tid[i] = CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]];
s.t. ctr15{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]];
s.t. ctr16{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]] * 666;
s.t. ctr17{i in CORE_LineItem2}: CORE_LineItem2quant[i] >= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]] - (1 -
CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]]) * 666;
s.t. ctr18{i in CORE_S}: CORE_Stid[i] <= sum{j in CORE_LineItem2} adj_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2tid[j];
s.t. ctr19{i in CORE_S, j in CORE_LineItem2}: CORE_Stid[i] >= adj_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2tid[j];
s.t. ctr20{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2quant[i];
s.t. ctr21{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2tid[i] * 666;
```

MathProg
or AMPL

Overview

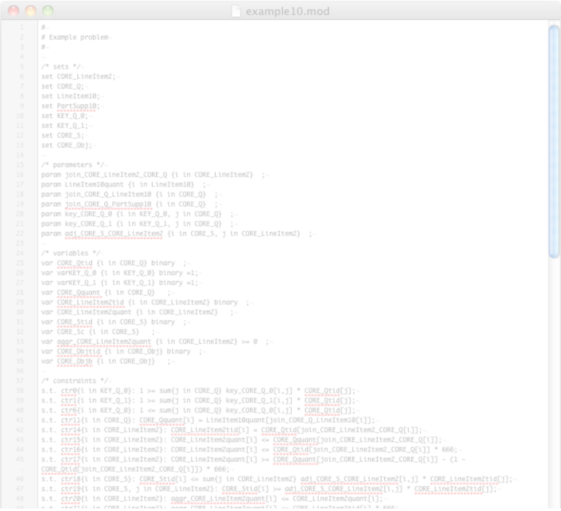


Visualizations

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)
```

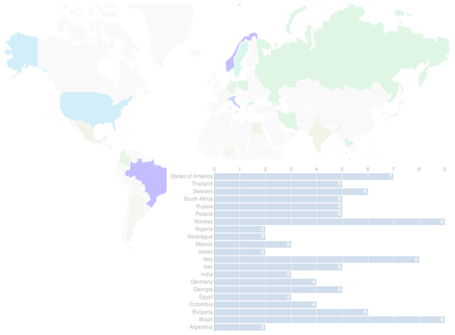
TiQL

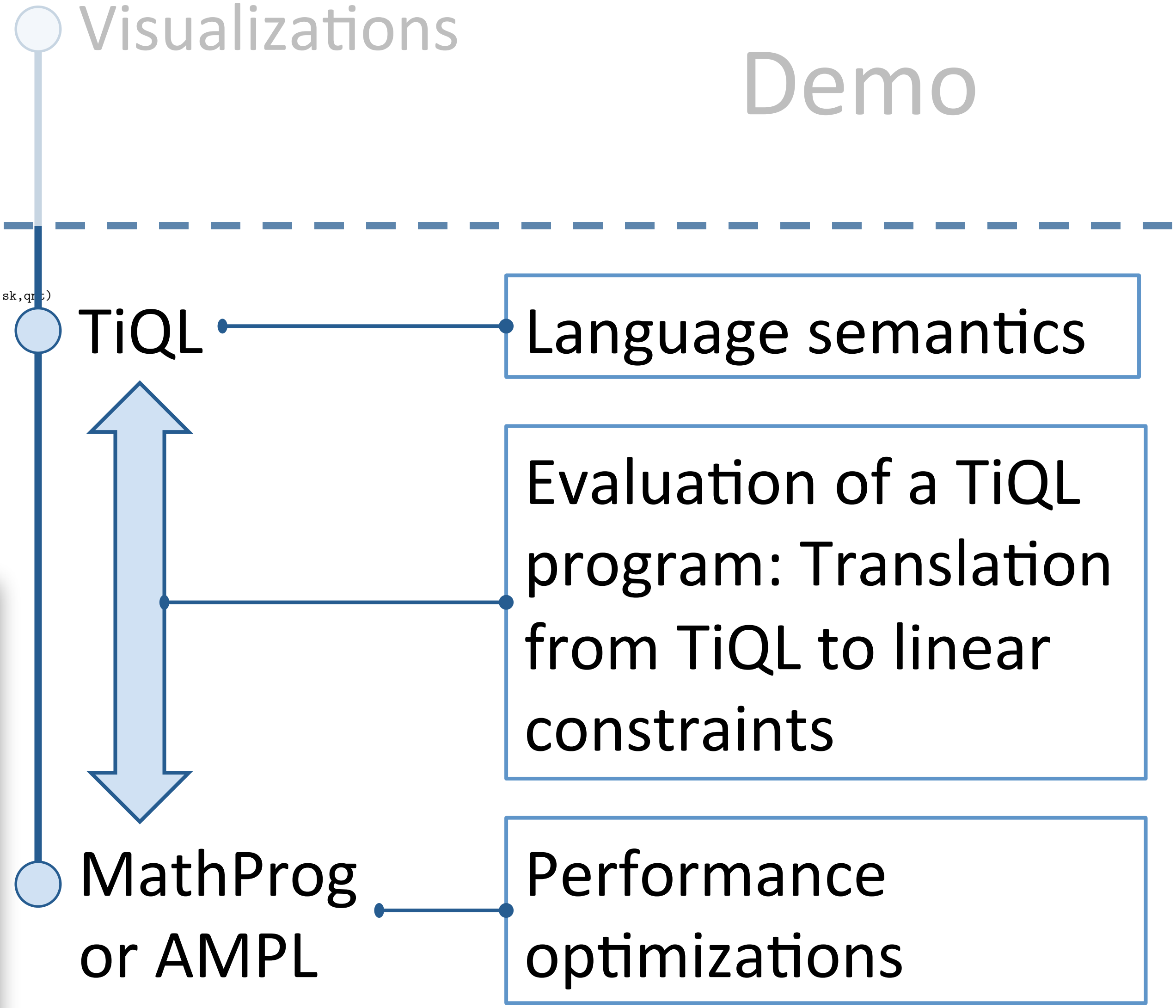


```
#
# Example problem
#

/* sets */
set CORE_LineItem2;
set CORE_Q;
set LineItem10;
set PartSupp10;
set KEY_Q_0;
set KEY_Q_1;
set CORE_S;
set CORE_Obj;

/* parameters */
param join_CORE_LineItem2_CORE_Q {i in CORE_LineItem2}  ;
param LineItem10quant {i in LineItem10}  ;
param join_CORE_Q_LineItem10 {i in CORE_Q}  ;
param join_CORE_Q_PartSupp10 {i in CORE_Q}  ;
param key_CORE_Q_0 {i in KEY_Q_0, j in CORE_Q}  ;
param key_CORE_Q_1 {i in KEY_Q_1, j in CORE_Q}  ;
param adj_CORE_S_CORE_LineItem2 {i in CORE_S, j in CORE_LineItem2}  ;

/* variables */
var CORE_Qtid {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_LineItem2tid {i in CORE_LineItem2} binary  ;
var CORE_LineItem2quant {i in CORE_LineItem2}   ;
var CORE_Stid {i in CORE_S} binary  ;
var CORE_Sc {i in CORE_S}   ;
var aggr_CORE_LineItem2quant {i in CORE_LineItem2} >= 0  ;
var CORE_Objtid {i in CORE_Obj} binary  ;
var CORE_Objb {i in CORE_Obj}   ;

/* constraints */
s.t. ctr0{i in KEY_Q_0}: 1 >= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr1{i in KEY_Q_1}: 1 >= sum{j in CORE_Q} key_CORE_Q_1[i,j] * CORE_Qtid[j];
s.t. ctr6{i in KEY_Q_0}: 1 <= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr11{i in CORE_Q}: CORE_Qquant[i] = LineItem10quant[join_CORE_Q_LineItem10[i]];
s.t. ctr14{i in CORE_LineItem2}: CORE_LineItem2tid[i] = CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]];
s.t. ctr15{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]];
s.t. ctr16{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]] * 666;
s.t. ctr17{i in CORE_LineItem2}: CORE_LineItem2quant[i] >= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]] - (1 -
CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]]) * 666;
s.t. ctr18{i in CORE_S}: CORE_Stid[i] <= sum{j in CORE_LineItem2} adj_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2tid[j];
s.t. ctr19{i in CORE_S, j in CORE_LineItem2}: CORE_Stid[i] >= adj_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2tid[j];
s.t. ctr20{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2quant[i];
s.t. ctr21{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2tid[i] * 666;
```

MathProg
or AMPL

Overview





Visualizations

Demo

TiQL

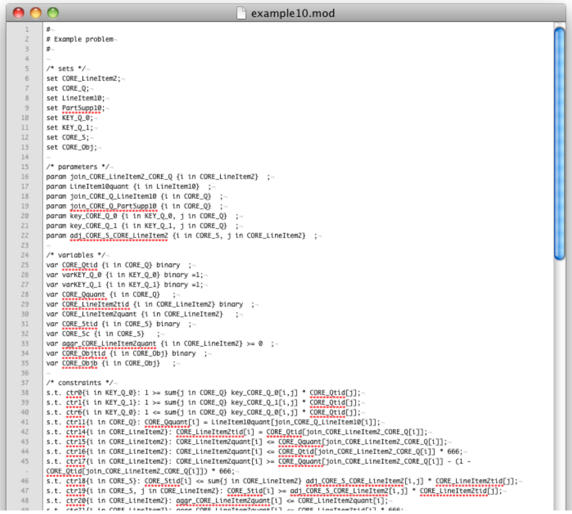
Language semantics

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

MathProg or AMPL

Performance optimizations

```
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)
```



```
#
# Example problem
#

/* sets */
set CORE_LineItem2;
set CORE_Q;
set LineItem10;
set PartSupp10;
set KEY_Q_0;
set KEY_Q_1;
set CORE_S;
set CORE_Obj;

/* parameters */
param join_CORE_LineItem2_CORE_Q {i in CORE_LineItem2}  ;
param LineItem10quant {i in LineItem10}  ;
param join_CORE_Q_LineItem10 {i in CORE_Q}  ;
param join_CORE_Q_PartSupp10 {i in CORE_Q}  ;
param key_CORE_Q_0 {i in KEY_Q_0, j in CORE_Q}  ;
param key_CORE_Q_1 {i in KEY_Q_1, j in CORE_Q}  ;
param adj_CORE_S_CORE_LineItem2 {i in CORE_S, j in CORE_LineItem2}  ;

/* variables */
var CORE_Qtid {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_LineItem2tid {i in CORE_LineItem2} binary  ;
var CORE_LineItem2quant {i in CORE_LineItem2}   ;
var CORE_Stid {i in CORE_S} binary  ;
var CORE_Sc {i in CORE_S}   ;
var aggr_CORE_LineItem2quant {i in CORE_LineItem2} >= 0  ;
var CORE_Objtid {i in CORE_Obj} binary  ;
var CORE_Objb {i in CORE_Obj}   ;

/* constraints */
s.t. ctr0{i in KEY_Q_0}: 1 >= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr1{i in KEY_Q_1}: 1 >= sum{j in CORE_Q} key_CORE_Q_1[i,j] * CORE_Qtid[j];
s.t. ctr6{i in KEY_Q_0}: 1 <= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr11{i in CORE_Q}: CORE_Qquant[i] = LineItem10quant[join_CORE_Q_LineItem10[i]];
s.t. ctr14{i in CORE_LineItem2}: CORE_LineItem2tid[i] = CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]];
s.t. ctr15{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]];
s.t. ctr16{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]] * 666;
s.t. ctr17{i in CORE_LineItem2}: CORE_LineItem2quant[i] >= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]] - (1 - CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]]) * 666;
s.t. ctr18{i in CORE_S}: CORE_Stid[i] <= sum{j in CORE_LineItem2} adj_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2tid[j];
s.t. ctr19{i in CORE_S, j in CORE_LineItem2}: CORE_Stid[i] >= adj_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2tid[j];
s.t. ctr20{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2quant[i];
s.t. ctr21{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2tid[i] * 666;
```

Tiresias Query Language

- Datalog-like notation:

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

- TiQL semantics:

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

$$\text{HDB} \longrightarrow HP(\bar{x}) \;:\!-\; body$$

TiQL Rules

Deduction Rule $\mathrm{HP}(\bar{x})$ `:- body`

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

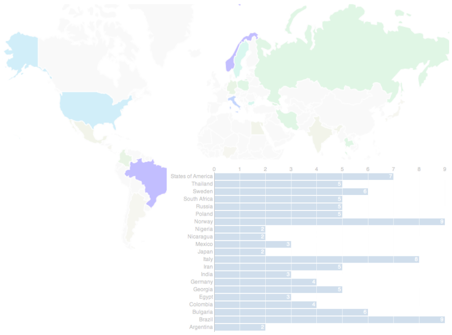
Reduction Rule $\mathrm{HP}(\bar{x})$ `:< body`

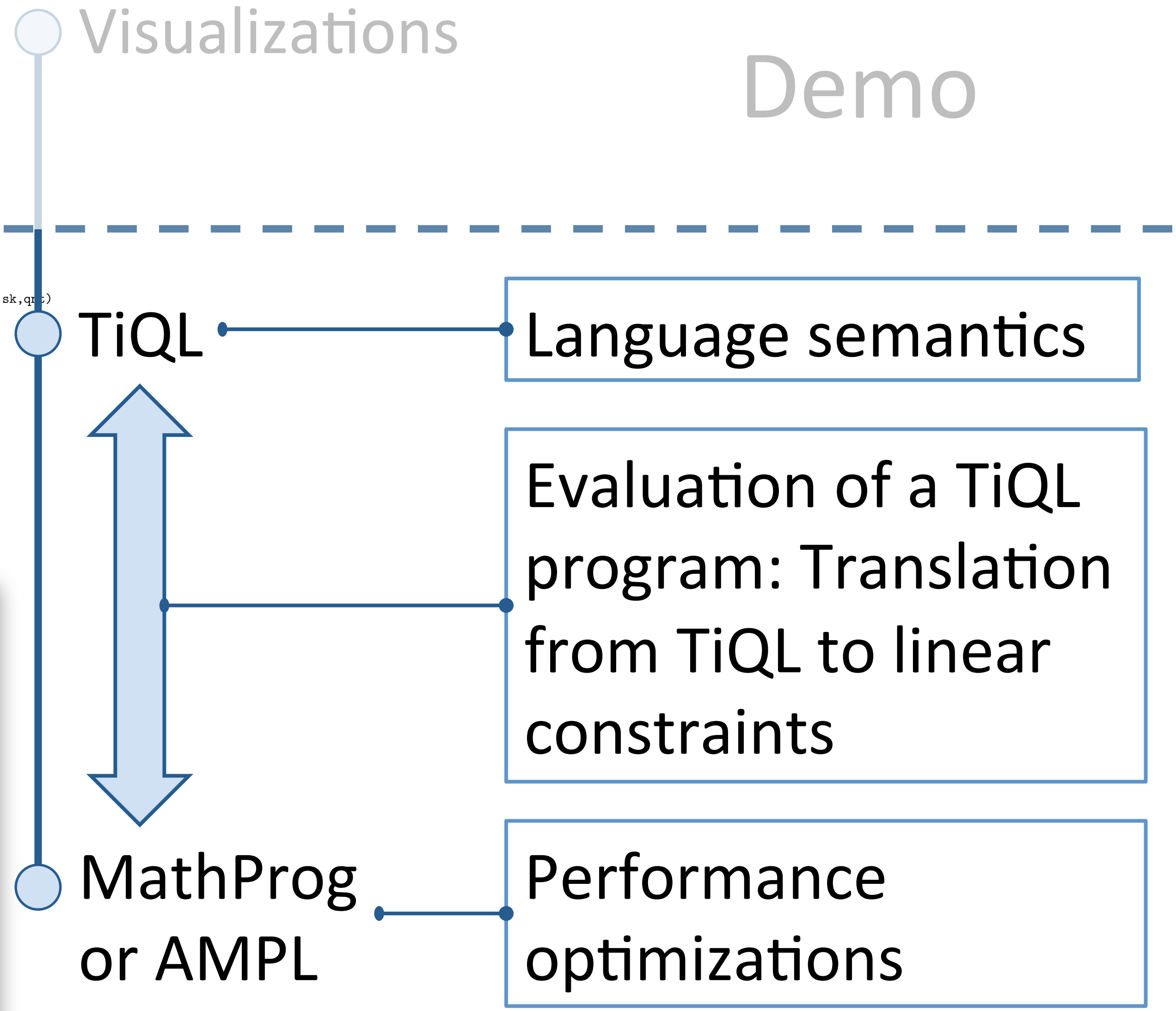
Semantics:
Takes a subset of tuples

Constraint Rule `[arithm-pred] <- body`

Semantics:
The head predicate needs to hold for all tuples

Overview





Visualizations

Demo

TiQL

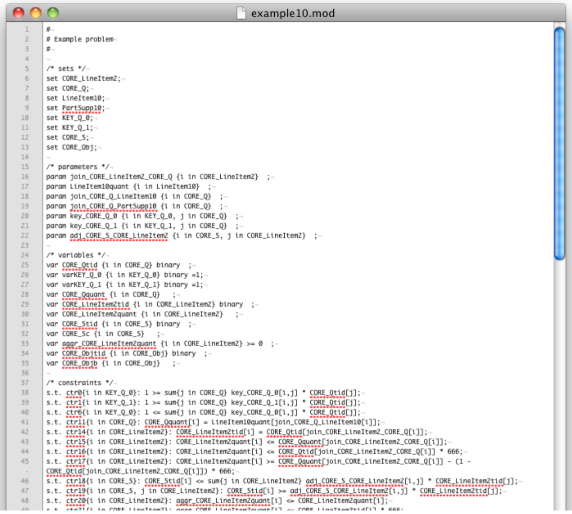
Language semantics

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

MathProg or AMPL

Performance optimizations

```
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)
```

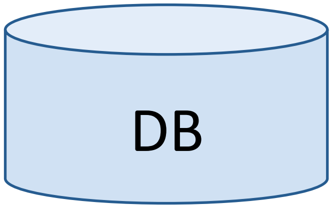


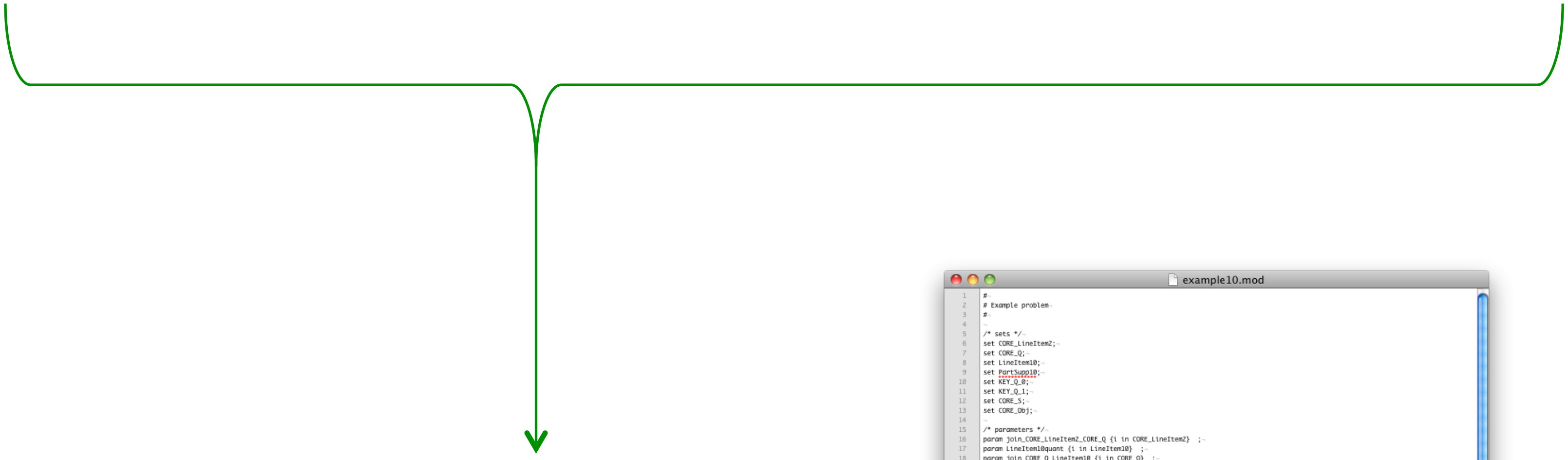
```
#
# Example problem
#

/* sets */
set CORE_LineItem2;
set CORE_Q;
set LineItem10;
set PartSupp10;
set KEY_Q_0;
set KEY_Q_1;
set CORE_S;
set CORE_Obj;

/* parameters */
param join_CORE_LineItem2_CORE_Q {i in CORE_LineItem2}  ;
param LineItem10quant {i in LineItem10}  ;
param join_CORE_Q_LineItem10 {i in CORE_Q}  ;
param join_CORE_Q_PartSupp10 {i in CORE_Q}  ;
param key_CORE_Q_0 {i in KEY_Q_0, j in CORE_Q}  ;
param key_CORE_Q_1 {i in KEY_Q_1, j in CORE_Q}  ;
param adj_CORE_S_CORE_LineItem2 {i in CORE_S, j in CORE_LineItem2}  ;

/* variables */
var CORE_Qtid {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_LineItem2tid {i in CORE_LineItem2} binary  ;
var CORE_LineItem2quant {i in CORE_LineItem2}   ;
var CORE_Stid {i in CORE_S} binary  ;
var CORE_Sc {i in CORE_S}   ;
var aggr_CORE_LineItem2quant {i in CORE_LineItem2} >= 0  ;
var CORE_Objtid {i in CORE_Obj} binary  ;
var CORE_Objb {i in CORE_Obj}   ;

/* constraints */
s.t. ctr0{i in KEY_Q_0}: 1 >= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr1{i in KEY_Q_1}: 1 >= sum{j in CORE_Q} key_CORE_Q_1[i,j] * CORE_Qtid[j];
s.t. ctr6{i in KEY_Q_0}: 1 <= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr11{i in CORE_Q}: CORE_Qquant[i] = LineItem10quant[join_CORE_Q_LineItem10[i]];
s.t. ctr14{i in CORE_LineItem2}: CORE_LineItem2tid[i] = CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]];
s.t. ctr15{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]];
s.t. ctr16{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]] * 666;
s.t. ctr17{i in CORE_LineItem2}: CORE_LineItem2quant[i] >= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]] - (1 -
CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]]) * 666;
s.t. ctr18{i in CORE_S}: CORE_Stid[i] <= sum{j in CORE_LineItem2} adj_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2tid[j];
s.t. ctr19{i in CORE_S, j in CORE_LineItem2}: CORE_Stid[i] >= adj_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2tid[j];
s.t. ctr20{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2quant[i];
s.t. ctr21{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2tid[i] * 666;
```

Evaluating a TiQL Program

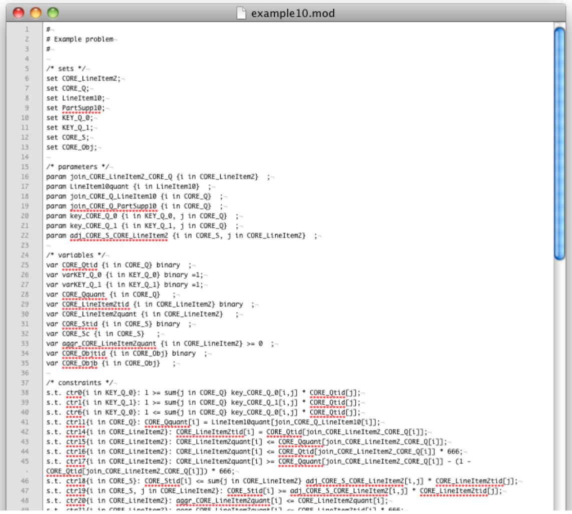
TiQL

```
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)
```





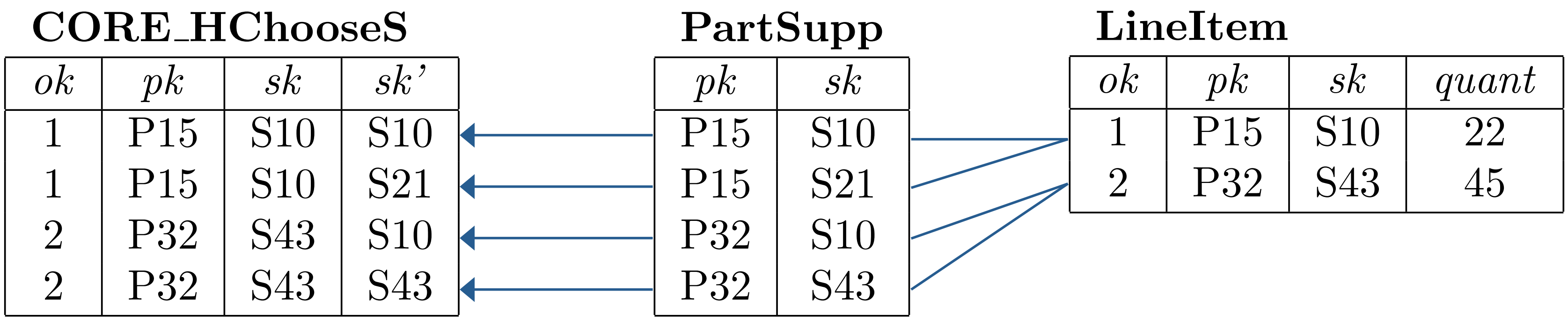
Mixed Integer Program
(MIP)



```
#
# Example problem
#

/* sets */
set CORE_LineItem2;
set CORE_Q;
set LineItem10;
set PartSupp10;
set KEY_Q_0;
set KEY_Q_1;
set CORE_S;
set CORE_Obj;

/* parameters */
param join_CORE_LineItem2_CORE_Q {i in CORE_LineItem2}  ;
param LineItem10quant {i in LineItem10}  ;
param join_CORE_Q_LineItem10 {i in CORE_Q}  ;
param join_CORE_Q_PartSupp10 {i in CORE_Q}  ;
param key_CORE_Q_0 {i in KEY_Q_0, j in CORE_Q}  ;
param key_CORE_Q_1 {i in KEY_Q_1, j in CORE_Q}  ;
param adj_CORE_S_CORE_LineItem2 {i in CORE_S, j in CORE_LineItem2}  ;

/* variables */
var CORE_Qtid {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_LineItem2tid {i in CORE_LineItem2} binary  ;
var CORE_LineItem2quant {i in CORE_LineItem2}   ;
var CORE_Stid {i in CORE_S} binary  ;
var CORE_Sc {i in CORE_S}   ;
var aggr_CORE_LineItem2quant {i in CORE_LineItem2} >= 0  ;
var CORE_Objtid {i in CORE_Obj} binary  ;
var CORE_Objb {i in CORE_Obj}   ;

/* constraints */
s.t. ctr0{i in KEY_Q_0}: 1 >= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr1{i in KEY_Q_1}: 1 >= sum{j in CORE_Q} key_CORE_Q_1[i,j] * CORE_Qtid[j];
s.t. ctr6{i in KEY_Q_0}: 1 <= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr11{i in CORE_Q}: CORE_Qquant[i] = LineItem10quant[join_CORE_Q_LineItem10[i]];
s.t. ctr14{i in CORE_LineItem2}: CORE_LineItem2tid[i] = CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]];
s.t. ctr15{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]];
s.t. ctr16{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]] * 666;
s.t. ctr17{i in CORE_LineItem2}: CORE_LineItem2quant[i] >= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]] - (1 - CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]]) * 666;
s.t. ctr18{i in CORE_S}: CORE_Stid[i] <= sum{j in CORE_LineItem2} adj_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2tid[j];
s.t. ctr19{i in CORE_S, j in CORE_LineItem2}: CORE_Stid[i] >= adj_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2tid[j];
s.t. ctr20{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2quant[i];
s.t. ctr21{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2tid[i] * 666;
```

Evaluating a TiQL Program

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



CORE_HChooseS

ok	*pk*	*sk*	*sk'*
1	P15	S10	S10
1	P15	S10	S21
2	P32	S43	S10
2	P32	S43	S43

PartSupp

pk	*sk*
P15	S10
P15	S21
P32	S10
P32	S43

LineItem

ok	*pk*	*sk*	*quant*
1	P15	S10	22
2	P32	S43	45

possible worlds

ok	*pk*	*sk*	*sk'*
1	P15	S10	S10
2	P32	S43	S43

ok	*pk*	*sk*	*sk'*
1	P15	S10	S21
2	P32	S43	S43

ok	*pk*	*sk*	*sk'*
1	P15	S10	S10
2	P32	S43	S10

ok	*pk*	*sk*	*sk'*
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

$Key(ok, pk, sk)$

$$x_1 + x_2 \leq 1$$

$$x_3 + x_4 \leq 1$$

$$0 \leq x_i \leq 1$$

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

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:

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

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

$$y \leq \sum_i x_i$$

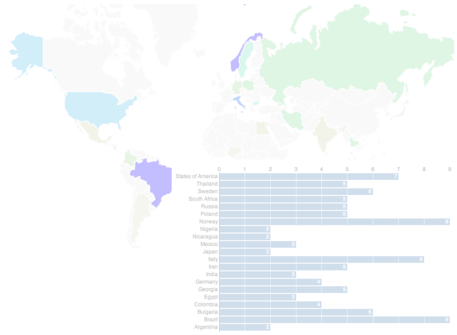
Conjunction:

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

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

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

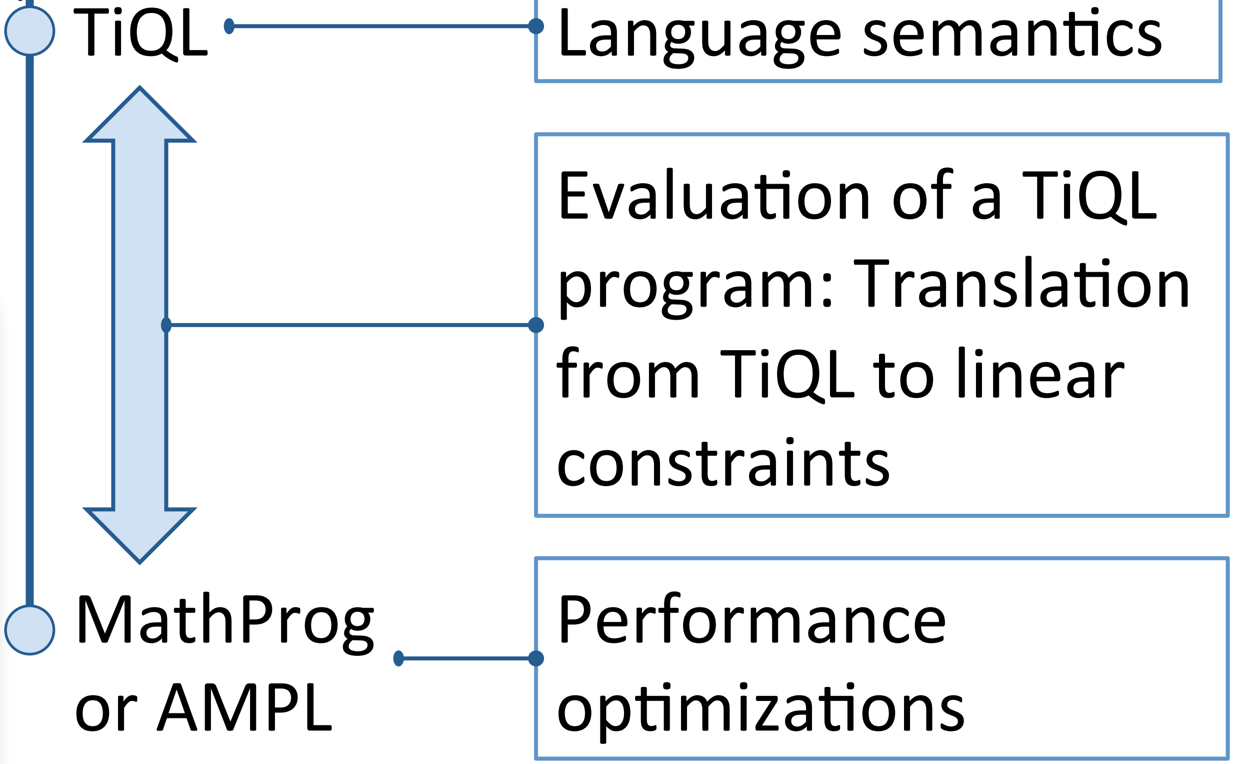
Overview



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)
```



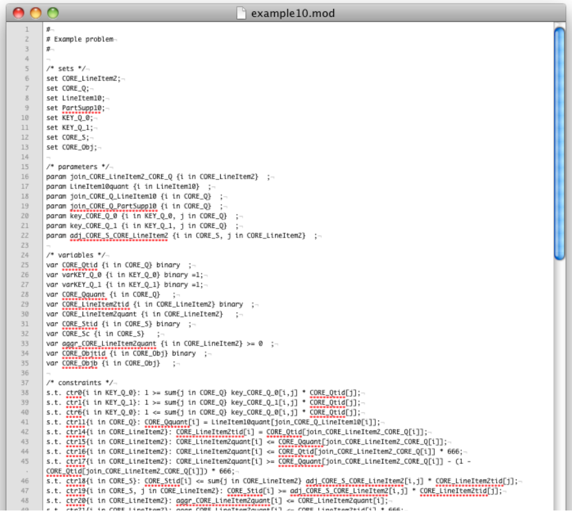
TiQL

Language semantics

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

MathProg or AMPL

Performance optimizations



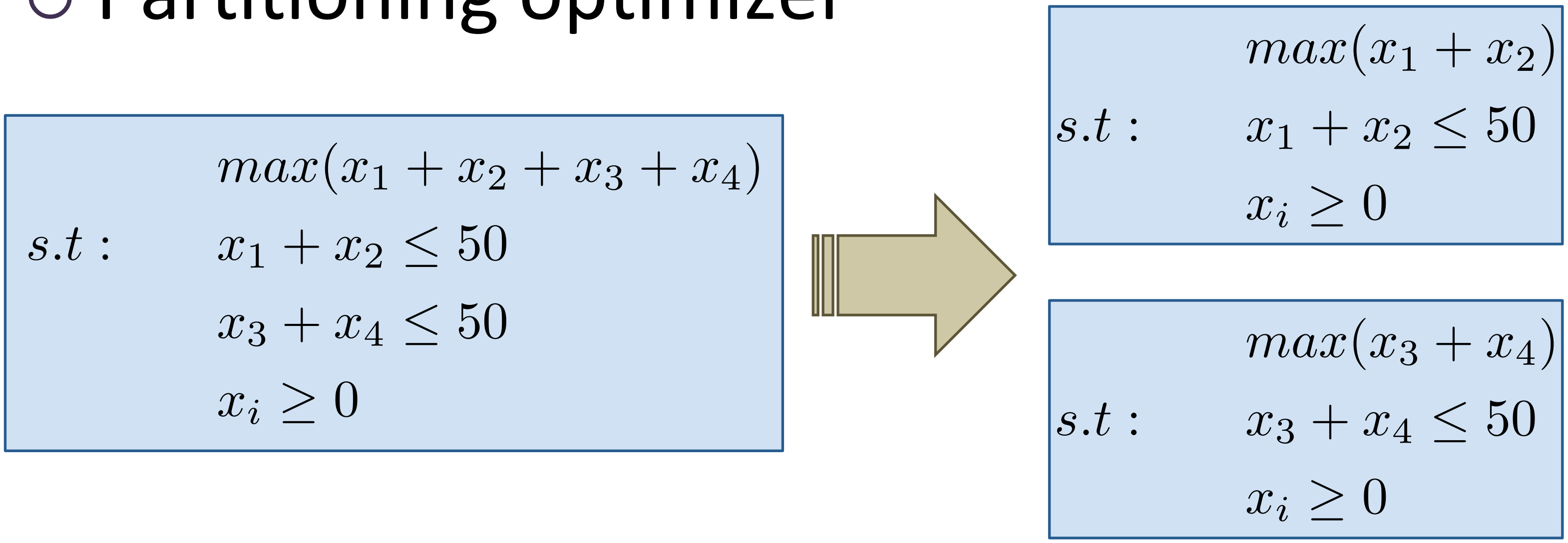
```
#
# Example problem
#

/* sets */
set CORE_LineItem2;
set CORE_Q;
set LineItem10;
set PartSupp10;
set KEY_Q_0;
set KEY_Q_1;
set CORE_S;
set CORE_Obj;

/* parameters */
param join_CORE_LineItem2_CORE_Q {i in CORE_LineItem2}  ;
param LineItem10quant {i in LineItem10}  ;
param join_CORE_Q_LineItem10 {i in CORE_Q}  ;
param join_CORE_Q_PartSupp10 {i in CORE_Q}  ;
param key_CORE_Q_0 {i in KEY_Q_0, j in CORE_Q}  ;
param key_CORE_Q_1 {i in KEY_Q_1, j in CORE_Q}  ;
param adj_CORE_S_CORE_LineItem2 {i in CORE_S, j in CORE_LineItem2}  ;

/* variables */
var CORE_Qtid {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_LineItem2tid {i in CORE_LineItem2} binary  ;
var CORE_LineItem2quant {i in CORE_LineItem2}   ;
var CORE_Stid {i in CORE_S} binary  ;
var CORE_Sc {i in CORE_S}   ;
var aggr_CORE_LineItem2quant {i in CORE_LineItem2} >= 0  ;
var CORE_Objtid {i in CORE_Obj} binary  ;
var CORE_Objb {i in CORE_Obj}   ;

/* constraints */
s.t. ctr0{i in KEY_Q_0}: 1 >= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr1{i in KEY_Q_1}: 1 >= sum{j in CORE_Q} key_CORE_Q_1[i,j] * CORE_Qtid[j];
s.t. ctr6{i in KEY_Q_0}: 1 <= sum{j in CORE_Q} key_CORE_Q_0[i,j] * CORE_Qtid[j];
s.t. ctr11{i in CORE_Q}: CORE_Qquant[i] = LineItem10quant[join_CORE_Q_LineItem10[i]];
s.t. ctr14{i in CORE_LineItem2}: CORE_LineItem2tid[i] = CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]];
s.t. ctr15{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]];
s.t. ctr16{i in CORE_LineItem2}: CORE_LineItem2quant[i] <= CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]] * 666;
s.t. ctr17{i in CORE_LineItem2}: CORE_LineItem2quant[i] >= CORE_Qquant[join_CORE_LineItem2_CORE_Q[i]] - (1 -
CORE_Qtid[join_CORE_LineItem2_CORE_Q[i]]) * 666;
s.t. ctr18{i in CORE_S}: CORE_Stid[i] <= sum{j in CORE_LineItem2} adj_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2tid[j];
s.t. ctr19{i in CORE_S, j in CORE_LineItem2}: CORE_Stid[i] >= adj_CORE_S_CORE_LineItem2[i,j] * CORE_LineItem2tid[j];
s.t. ctr20{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2quant[i];
s.t. ctr21{i in CORE_LineItem2}: aggr_CORE_LineItem2quant[i] <= CORE_LineItem2tid[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

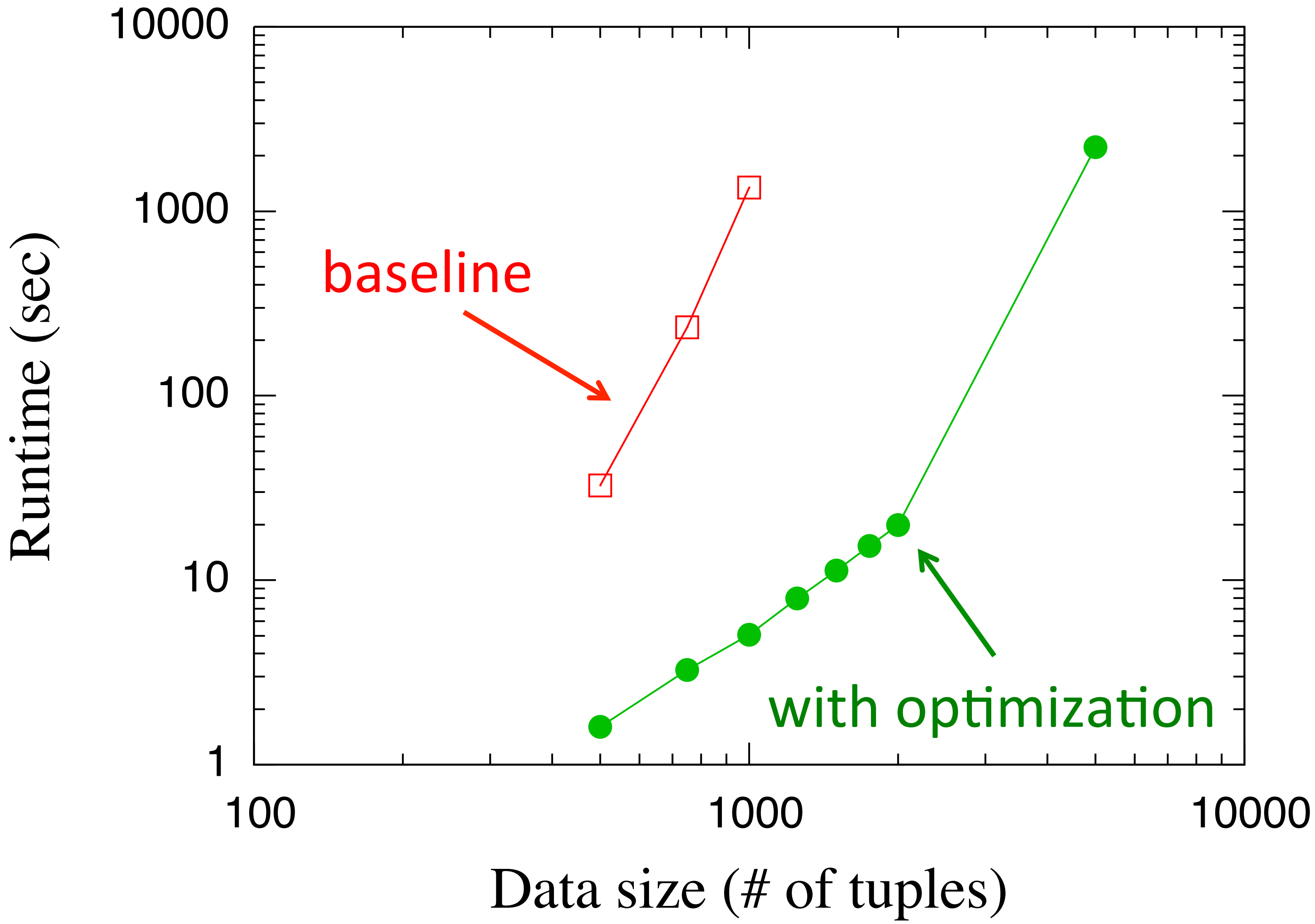


$$\begin{aligned} & max(x_1 + x_2 + x_3 + x_4) \\ s.t: \quad & x_1 + x_2 \leq 50 \\ & x_3 + x_4 \leq 50 \\ & x_i \geq 0 \end{aligned}$$

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

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

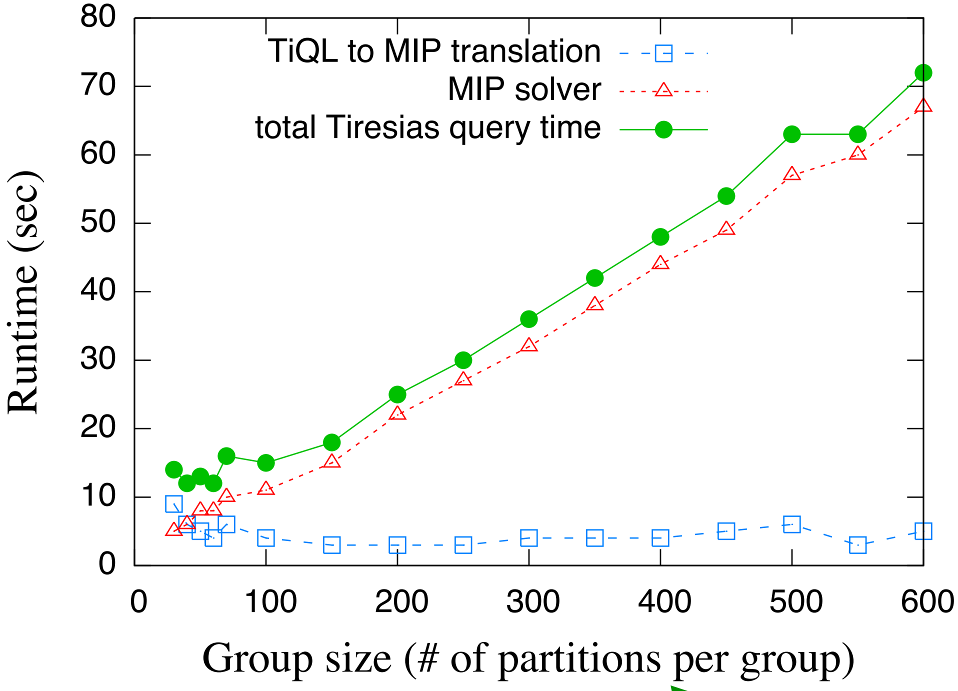
Evaluation of the Model Optimizer



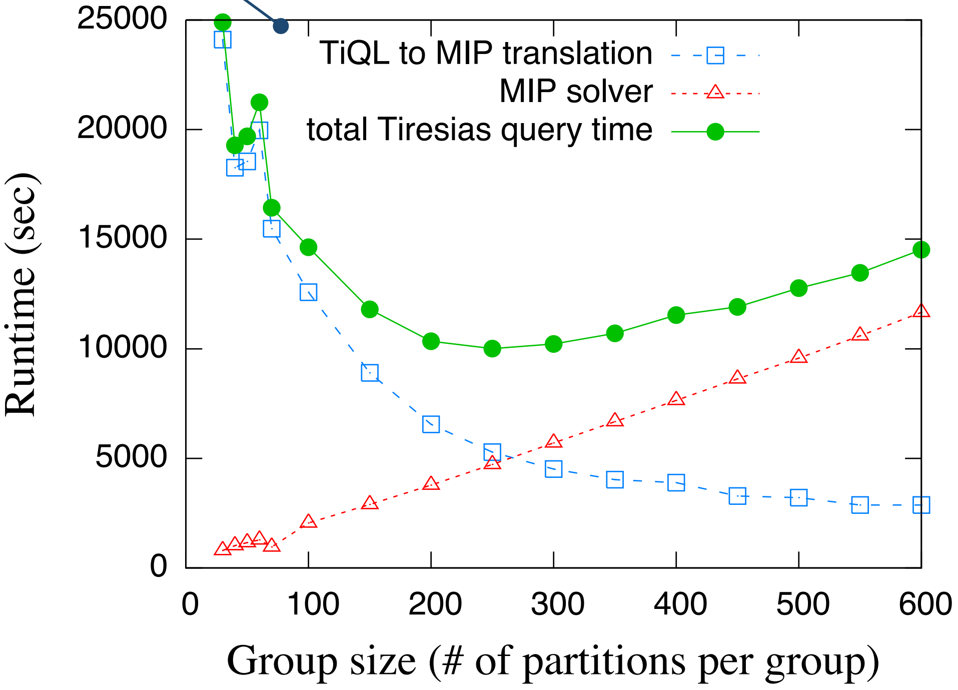
Evaluation of Tiresias Partitioning

complex dependency on the granularity of partitioning

10k tuples

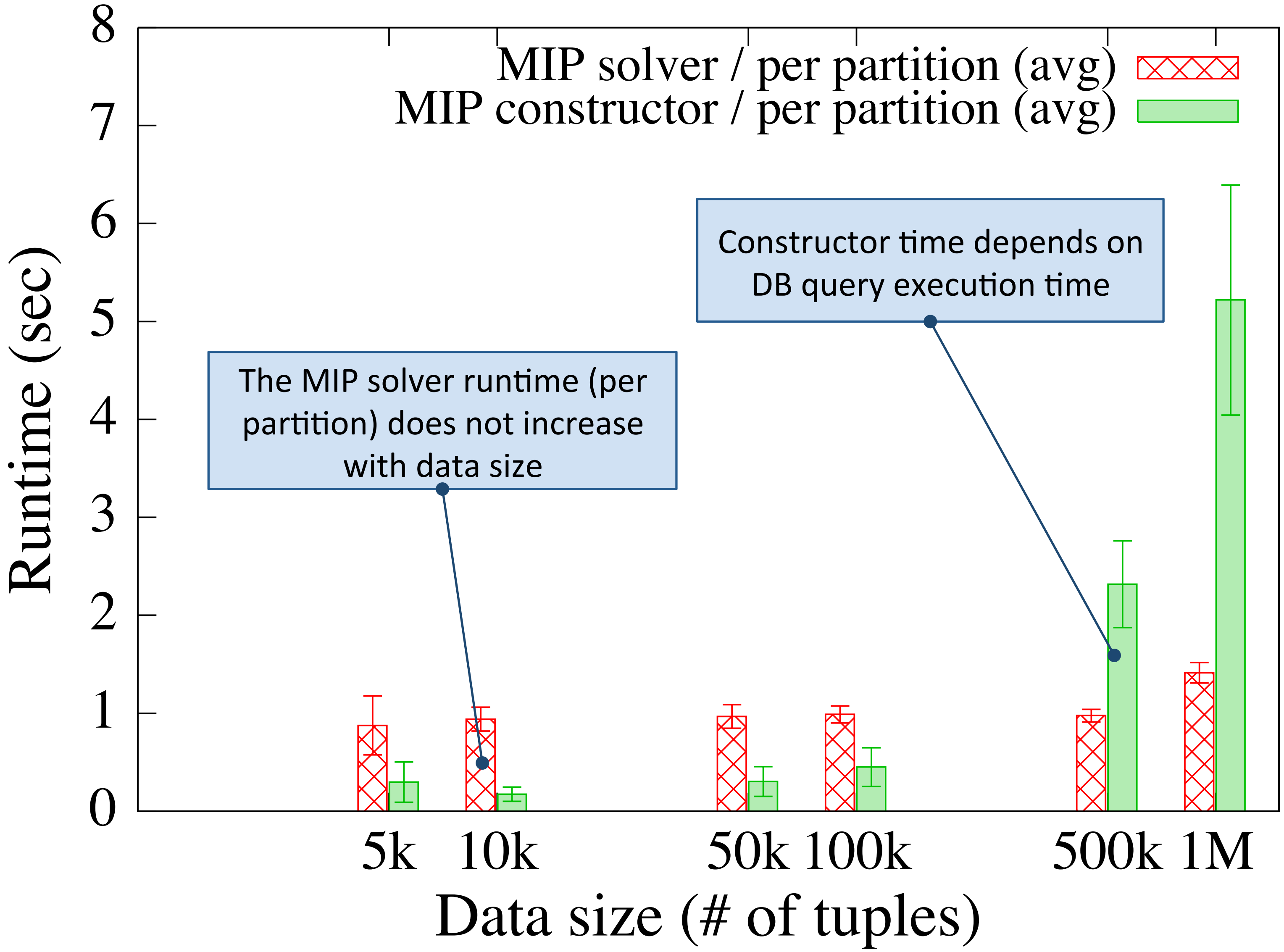


1M tuples



granularity of partitioning

Scalability



The MIP solver runtime (per partition) does not increase with data size

Constructor time depends on DB query execution time

Related Work

- Provenance

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

- Incomplete databases

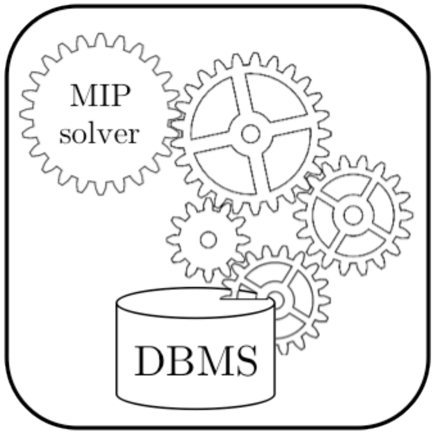
  [Antonova et al. SIGMOD’07], [Imielinski et al. JACM’84], [Koch ICDT’09]

- 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

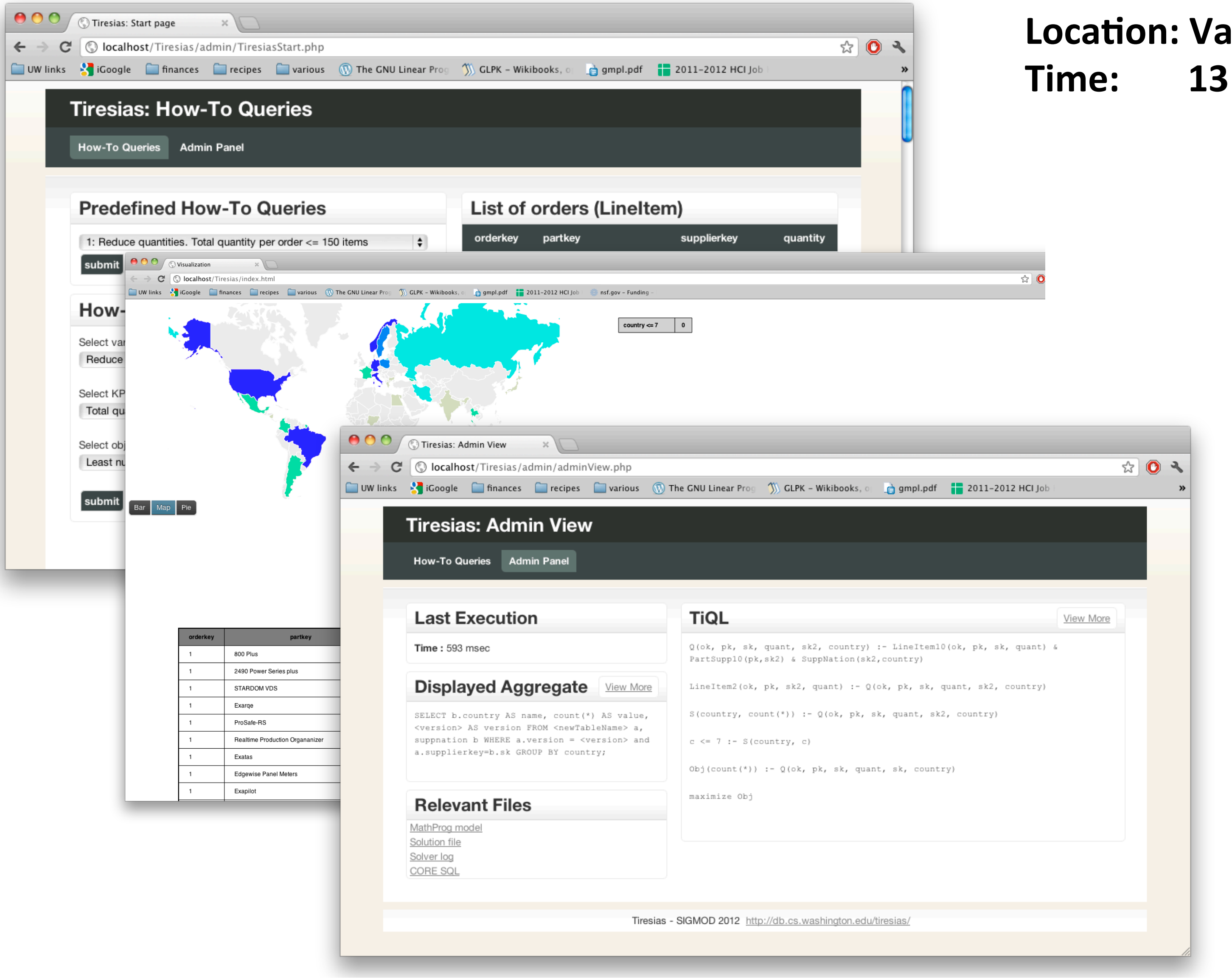
Approximations

Parallelization and handling of skew

Result analysis and feedback-based problem generation

SIGMOD Demo Group C

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



Tiresias: How-To Queries

How-To Queries | Admin Panel

Predefined How-To Queries

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

submit

How-

Select var
Reduce

Select KP
Total qu

Select obj
Least nu

submit

List of orders (LineItem)

orderkey	partkey	supplierkey	quantity

orderkey	partkey
1	800 Plus
1	2490 Power Series plus
1	STARDOM VDS
1	Exarqe
1	ProSafe-RS
1	Realtime Production Organanizer
1	Exatas
1	Edgewise Panel Meters
1	Exapilot

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

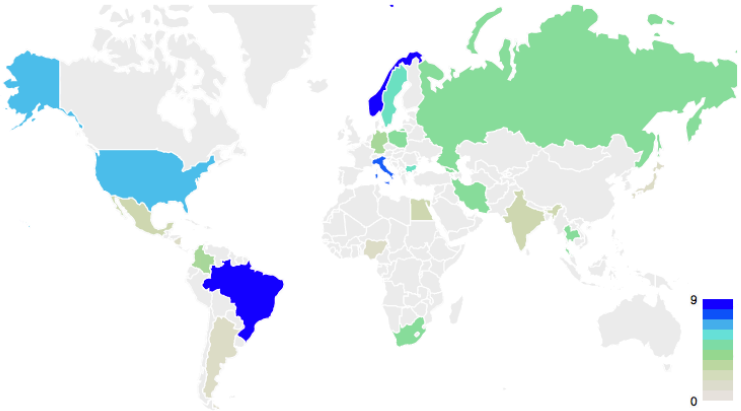
View More

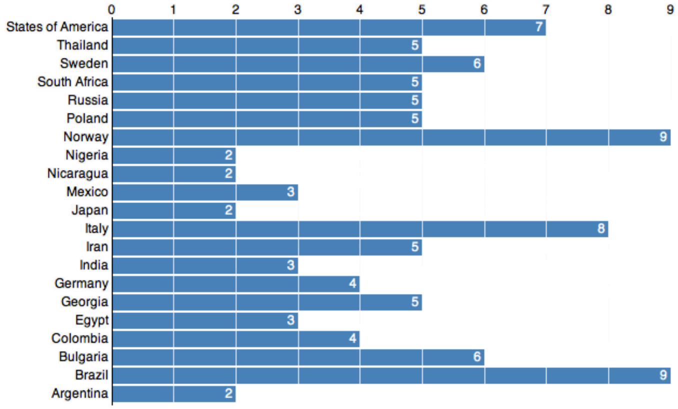
```
Q(ok, pk, sk, quant, sk2, country) :- LineItem10(ok, pk, sk, quant) &
PartSupp10(pk,sk2) & SuppNation(sk2,country)

LineItem2(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