

# Range-Consistent Answers of Aggregate Queries under Aggregate Constraints

Sergio Flesca, Filippo Furfaro, **Francesco Parisi**

DEIS  
University of Calabria  
87036 Rende (CS), Italy

SUM 2010  
Toulouse, September, 27 - 29

# Inconsistent Numerical Data

- Data inconsistency can arise in several scenarios
  - Data integration, reconciliation, errors in acquiring data (mistakes in transcription, OCR tools, sensors, etc.)
- Acquiring balance sheets data

original (consistent)  
 balance-sheet  
 paper document

Receipts	cash sales	100
	receivables	120
	total receipts	220

- The original data were **consistent**:  $100 + 120 = 220$ , but a symbol recognition error occurred during the digitizing phase

digitized document  
 (e.g. obtained by an OCR tool)

Receipts	cash sales	100
	receivables	120
	total receipts	250

- The acquired document is **not consistent**:  $100 + 120 \neq 250$

# Inconsistent Numerical Data

- Data inconsistency can arise in several scenarios
  - Data integration, reconciliation, errors in acquiring data (mistakes in transcription, OCR tools, sensors, etc.)
- Acquiring balance sheets data

original (consistent)  
 balance-sheet  
 paper document

Receipts	cash sales	100
	receivables	120
	total receipts	220

- The original data were **consistent**:  $100 + 120 = 220$ , but a symbol recognition error occurred during the digitizing phase

digitized document  
 (e.g. obtained by an OCR tool)

Receipts	cash sales	100
	receivables	120
	total receipts	250

- The acquired document is **not consistent**:  $100 + 120 \neq 250$

# Inconsistent Numerical Data

- Data inconsistency can arise in several scenarios
  - Data integration, reconciliation, errors in acquiring data (mistakes in transcription, OCR tools, sensors, etc.)
- Acquiring balance sheets data

original (consistent)  
 balance-sheet  
 paper document

Receipts	cash sales	100
	receivables	120
	total receipts	220

- The original data were **consistent**:  $100 + 120 = 220$ , but a symbol recognition error occurred during the digitizing phase

digitized document  
 (e.g. obtained by an OCR tool)

Receipts	cash sales	100
	receivables	120
	total receipts	250

- The acquired document is **not consistent**:  $100 + 120 \neq 250$

# Querying Inconsistent Data

- The analysis of the financial conditions of a company can be supported by evaluating **aggregate queries** on its (digitized) balance sheets
- Examples of queries which can support this kind of analysis are:
  - the maximum/minimum value of *cash sales* over the last five years
  - the sum of *cash sales* for the last five years
- The mere evaluation of these queries on inconsistent data may yield a wrong picture of the real world

# Querying Inconsistent Data

- The analysis of the financial conditions of a company can be supported by evaluating **aggregate queries** on its (digitized) balance sheets
- Examples of queries which can support this kind of analysis are:
  - the maximum/minimum value of *cash sales* over the last five years
  - the sum of *cash sales* for the last five years
- The mere evaluation of these queries on inconsistent data may yield a wrong picture of the real world

# Querying Inconsistent Data

- The analysis of the financial conditions of a company can be supported by evaluating **aggregate queries** on its (digitized) balance sheets
- Examples of queries which can support this kind of analysis are:
  - the maximum/minimum value of *cash sales* over the last five years
  - the sum of *cash sales* for the last five years
- The mere evaluation of these queries on inconsistent data may yield a wrong picture of the real world

# Range-Consistent Answers (Range-CQAs)

- The **range-consistent answer** of an aggregate query is the narrowest interval containing all the answers of the query evaluated on every possible repaired database
- Range-CQAs can still support several analysis tasks
- For instance, knowing that, for every “reasonable” repair,
  - the maximum and the minimum of *cash sales* are in the intervals  $[100, 120]$  and  $[50, 70]$ , respectively,
  - the sum of *cash sales* for the considered years is in  $[350, 400]$can give a **sufficiently accurate picture** of the trend of cash sales.



# Range-Consistent Answers (Range-CQAs)

- The **range-consistent answer** of an aggregate query is the narrowest interval containing all the answers of the query evaluated on every possible repaired database
- Range-CQAs can still support several analysis tasks
- For instance, knowing that, for every “reasonable” repair,
  - the maximum and the minimum of *cash sales* are in the intervals  $[100, 120]$  and  $[50, 70]$ , respectively,
  - the sum of *cash sales* for the considered years is in  $[350, 400]$can give a **sufficiently accurate picture** of the trend of cash sales.

# Range-CQAs under Aggregate Constraints

- We devised a strategy for computing range consistent answers of SUM-, MIN-, and MAX-queries in the presence of aggregate constraints
- Our approach computes range-CQAs by solving Integer Linear Programming (ILP) problem instances, thus enabling the computation of range-CQAs by means of well-known techniques for solving ILP
- We characterized the computational complexity of the range-CQA problem for SUM-, MIN-, and MAX-queries in the presence of aggregate constraints
- We experimentally validated our approach

# Range-CQAs under Aggregate Constraints

- We devised a strategy for computing range consistent answers of  $SUM$ -,  $MIN$ -, and  $MAX$ -queries in the presence of aggregate constraints
- Our approach computes range-CQAs by solving Integer Linear Programming (ILP) problem instances, thus enabling the computation of range-CQAs by means of well-known techniques for solving ILP
- We characterized the computational complexity of the range-CQA problem for  $SUM$ -,  $MIN$ -, and  $MAX$ -queries in the presence of aggregate constraints
- We experimentally validated our approach

# Range-CQAs under Aggregate Constraints

- We devised a strategy for computing range consistent answers of  $SUM$ -,  $MIN$ -, and  $MAX$ -queries in the presence of aggregate constraints
- Our approach computes range-CQAs by solving Integer Linear Programming (ILP) problem instances, thus enabling the computation of range-CQAs by means of well-known techniques for solving ILP
- We characterized the computational complexity of the range-CQA problem for  $SUM$ -,  $MIN$ -, and  $MAX$ -queries in the presence of aggregate constraints
- We experimentally validated our approach

# Range-CQAs under Aggregate Constraints

- We devised a strategy for computing range consistent answers of  $SUM$ -,  $MIN$ -, and  $MAX$ -queries in the presence of aggregate constraints
- Our approach computes range-CQAs by solving Integer Linear Programming (ILP) problem instances, thus enabling the computation of range-CQAs by means of well-known techniques for solving ILP
- We characterized the computational complexity of the range-CQA problem for  $SUM$ -,  $MIN$ -, and  $MAX$ -queries in the presence of aggregate constraints
- We experimentally validated our approach

# Outline

- 1 Introduction
  - Motivation
  - Contribution
- 2 **Preliminaries**
  - **Aggregate Constraints**
  - **Repairs**
  - **Aggregate Queries**
- 3 Query Answering
  - Steady Aggregate Constraints
  - Computing Range-Consistent Answers
  - Experimental Results
- 4 Conclusion and Future Work

# Managing data consistency

- Often classical “classical” integrity constraints (keys, foreign keys, FDs) do not suffice to manage data consistency
  - in scientific and statistical databases, data warehouses, numerical values in some tuples result from **aggregating values** in other tuples
  - in the balance sheet example, *the sum of cash sales and receivables should be equal to the total cash receipts*

digitized document  
(e.g. obtained by an OCR tool)

Receipts	cash sales	100
	receivables	120
	total receipts	250

- **Aggregate constraints** allow us to define algebraic relations among aggregate values extracted from the database

# Managing data consistency

- Often classical “classical” integrity constraints (keys, foreign keys, FDs) do not suffice to manage data consistency
  - in scientific and statistical databases, data warehouses, numerical values in some tuples result from **aggregating values** in other tuples
  - in the balance sheet example, *the sum of cash sales and receivables should be equal to the total cash receipts*

digitized document  
 (e.g. obtained by an OCR tool)

Receipts	cash sales	100
	receivables	120
	total receipts	250

- **Aggregate constraints** allow us to define algebraic relations among aggregate values extracted from the database



# Managing data consistency

- Often classical “classical” integrity constraints (keys, foreign keys, FDs) do not suffice to manage data consistency
  - in scientific and statistical databases, data warehouses, numerical values in some tuples result from **aggregating values** in other tuples
  - in the balance sheet example, *the sum of cash sales and receivables should be equal to the total cash receipts*

digitized document  
(e.g. obtained by an OCR tool)

Receipts	cash sales	100
	receivables	120
	total receipts	250

- **Aggregate constraints** allow us to define algebraic relations among aggregate values extracted from the database

# Aggregate Constraints

## Definition (Aggregate Constraint)

An aggregate constraint on a database scheme  $\mathcal{D}$  is of the form

$$\forall \vec{x} \left( \phi(\vec{x}) \implies \sum_{i=1}^n c_i \cdot \chi_i(\vec{y}_i) \leq K \right)$$

- 1  $c_1, \dots, c_n, K$  are rational constants;
  - 2  $\phi(\vec{x})$  is a conjunction of atoms constructed from relation names, constants, and all the variables in  $\vec{x}$ ;
  - 3 each  $\chi_i(\vec{y}_i)$  is an aggregation function, where  $\vec{y}_i$  is a list of variables and constants, and every variable that occurs in  $\vec{y}_i$  also occurs in  $\vec{x}$ .
- The aggregation function  $\chi(\vec{y}) = \langle R, e, \alpha(\vec{y}) \rangle$  corresponds to the SQL query `SELECT SUM (e) FROM R WHERE  $\alpha(\vec{y})$` , where  $e$  is an attribute of  $R$  or a constant

# Aggregate Constraints

## Definition (Aggregate Constraint)

An aggregate constraint on a database scheme  $\mathcal{D}$  is of the form

$$\forall \vec{x} \left( \phi(\vec{x}) \implies \sum_{i=1}^n c_i \cdot \chi_i(\vec{y}_i) \leq K \right)$$

- 1  $c_1, \dots, c_n, K$  are rational constants;
  - 2  $\phi(\vec{x})$  is a conjunction of atoms constructed from relation names, constants, and all the variables in  $\vec{x}$ ;
  - 3 each  $\chi_i(\vec{y}_i)$  is an aggregation function, where  $\vec{y}_i$  is a list of variables and constants, and every variable that occurs in  $\vec{y}_i$  also occurs in  $\vec{x}$ .
- The aggregation function  $\chi(\vec{y}) = \langle R, e, \alpha(\vec{y}) \rangle$  corresponds to the SQL query `SELECT SUM (e) FROM R WHERE  $\alpha(\vec{y})$` , where  $e$  is an attribute of  $R$  or a constant

# Example of Aggregate Constraint

## *BalanceSheets*

<i>Year</i>	<i>Section</i>	<i>Subsection</i>	<i>Type</i>	<i>Value</i>
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year

- $\chi_1(x, y, z) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year}=x \wedge \text{Section}=y \wedge \text{Type}=z) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, x_2, \text{'det'}) = \chi_1(x_1, x_2, \text{'aggr'})$

# Example of Aggregate Constraint

*BalanceSheets*

<i>Year</i>	<i>Section</i>	<i>Subsection</i>	<i>Type</i>	<i>Value</i>
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year

- $\chi_1(x, y, z) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year}=x \wedge \text{Section}=y \wedge \text{Type}=z) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, x_2, \text{'det'}) = \chi_1(x_1, x_2, \text{'aggr'})$

## Example of Aggregate Constraint

*BalanceSheets*

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year

- $\chi_1(x, y, z) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year}=x \wedge \text{Section}=y \wedge \text{Type}=z) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, x_2, \text{'det'}) = \chi_1(x_1, x_2, \text{'aggr'})$

## Repairing strategy (1/2)

- A repair for a database w.r.t. a set of aggregate constraints is a set of **value updates** making the database consistent
- Updates regard attributes representing measure values, such as weights, lengths, prices, etc. We call these attributes *measure attributes*
- We assume that the absolute values of measure attributes are **bounded by** a constant  $M$ .
  - It is often possible to pre-determine a specific range for numerical attributes.
  - In the balance sheet context, it can be reasonably assumed that the items are bounded by  $\$ 10^9$ .

## Repairing strategy (1/2)

- A repair for a database w.r.t. a set of aggregate constraints is a set of **value updates** making the database consistent
- Updates regard attributes representing measure values, such as weights, lengths, prices, etc. We call these attributes *measure attributes*
- We assume that the absolute values of measure attributes are **bounded by** a constant  $M$ .
  - It is often possible to pre-determine a specific range for numerical attributes.
  - In the balance sheet context, it can be reasonably assumed that the items are bounded by  $\$ 10^9$ .



## Repairing strategy (2/2)

- Reasonable repairs, called *card-minimal repairs*, are those having minimum cardinality
- Repairing by *card-minimal repairs* means assuming that the minimum number of errors occurred
  - In the balance-sheet context: the most probable case is that the acquiring system made the minimum number of errors

## Repairing strategy (2/2)

- Reasonable repairs, called *card-minimal repairs*, are those having minimum cardinality
- Repairing by *card-minimal repairs* means assuming that the minimum number of errors occurred
  - In the balance-sheet context: the most probable case is that the acquiring system made the minimum number of errors

## Two examples of *card*-minimal repairs

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$        $\rho_2$   
 $\rightarrow 130$        $\rightarrow 150$

- $\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year
- $\kappa_2$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- $\kappa_3$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*

## Two examples of *card*-minimal repairs

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$        $\rho_2$   
 $\rightarrow 130$        $\rightarrow 150$

- $\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year
- $\kappa_2$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- $\kappa_3$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*

## Two examples of *card*-minimal repairs

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$        $\rho_2$   
 $\rightarrow 130$        $\rightarrow 150$

- $\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year
- $\kappa_2$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- $\kappa_3$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*

## Two examples of *card*-minimal repairs

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$                        $\rho_2$   
 → 130                      → 150

- $\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year
- $\kappa_2$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- $\kappa_3$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*

# Aggregate Queries

## Definition (Aggregate Query)

An aggregate query on a database scheme  $\mathcal{D}$  is an expression of the form `SELECT  $f$  FROM  $R$  WHERE  $\alpha$` , where:

- 1  $R$  is a relation scheme in  $\mathcal{D}$ ;
- 2  $f$  is one of  $\text{MIN}(A)$ ,  $\text{MAX}(A)$  or  $\text{SUM}(A)$ , where  $A$  is an attribute of  $R$ ;
- 3  $\alpha$  is boolean combination of atomic comparisons of the form  $X \diamond Y$ , where  $X$  and  $Y$  are constants or non-measure attributes of  $R$ , and  $\diamond \in \{=, \neq, \leq, \geq, <, >\}$ .

- Our transformation for computing CQAs by solving ILP instances exploits the restriction that no measure attribute occurs in the `WHERE` clause of an aggregate query

# Aggregate Queries

## Definition (Aggregate Query)

An aggregate query on a database scheme  $\mathcal{D}$  is an expression of the form `SELECT  $f$  FROM  $R$  WHERE  $\alpha$` , where:

- 1  $R$  is a relation scheme in  $\mathcal{D}$ ;
  - 2  $f$  is one of  $\text{MIN}(A)$ ,  $\text{MAX}(A)$  or  $\text{SUM}(A)$ , where  $A$  is an attribute of  $R$ ;
  - 3  $\alpha$  is boolean combination of atomic comparisons of the form  $X \diamond Y$ , where  $X$  and  $Y$  are constants or non-measure attributes of  $R$ , and  $\diamond \in \{=, \neq, \leq, \geq, <, >\}$ .
- Our transformation for computing CQAs by solving ILP instances exploits the restriction that no measure attribute occurs in the `WHERE` clause of an aggregate query



## Range Consistent Answers

Let  $\mathcal{D}$  be a database scheme,  $\mathcal{AC}$  a set of aggregate constraints on  $\mathcal{D}$ ,  $q$  an aggregate query on  $\mathcal{D}$ , and  $D$  an instance of  $\mathcal{D}$ .

### Definition (Range-consistent query answer)

The *range-consistent query answer* of  $q$  on  $D$  is the empty interval  $\emptyset$ , in the case that  $D$  admits no repair w.r.t.  $\mathcal{AC}$ , or the interval  $[glb, lub]$ , otherwise, where:

- i) for each *card*-minimal repair  $\rho$  for  $D$  w.r.t.  $\mathcal{AC}$ , it holds that  $glb \leq q(\rho(D)) \leq lub$ ;
- ii) there is a pair  $\rho', \rho''$  of *card*-minimal repairs for  $D$  w.r.t.  $\mathcal{AC}$  such that  $q(\rho'(D)) = glb$  and  $q(\rho''(D)) = lub$ .

## Range Consistent Answers

Let  $\mathcal{D}$  be a database scheme,  $\mathcal{AC}$  a set of aggregate constraints on  $\mathcal{D}$ ,  $q$  an aggregate query on  $\mathcal{D}$ , and  $D$  an instance of  $\mathcal{D}$ .

### Definition (Range-consistent query answer)

The *range-consistent query answer* of  $q$  on  $D$  is the empty interval  $\emptyset$ , in the case that  $D$  admits no repair w.r.t.  $\mathcal{AC}$ , or the interval  $[glb, lub]$ , otherwise, where:

- i) for each *card*-minimal repair  $\rho$  for  $D$  w.r.t.  $\mathcal{AC}$ , it holds that  $glb \leq q(\rho(D)) \leq lub$ ;
- ii) there is a pair  $\rho', \rho''$  of *card*-minimal repairs for  $D$  w.r.t.  $\mathcal{AC}$  such that  $q(\rho'(D)) = glb$  and  $q(\rho''(D)) = lub$ .

# Range Consistent Answers - Example

## BalanceSheets

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$                        $\rho_2$   
 → 130                      → 150

- The range-CQA of `SELECT MAX(Value) FROM BalanceSheets WHERE Subsection = 'cash sales'` is [100, 130]
- The range-CQA of `SELECT MAX(Value) FROM BalanceSheets WHERE Subsection = 'net cash inflow'` is [30, 30]

# Range Consistent Answers - Example

## BalanceSheets

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$                        $\rho_2$   
→ 130                      → 150

- The range-CQA of  $\text{SELECT MAX}(\text{Value})$  FROM *BalanceSheets* WHERE *Subsection* = 'cash sales' is [100, 130]
- The range-CQA of  $\text{SELECT MAX}(\text{Value})$  FROM *BalanceSheets* WHERE *Subsection* = 'net cash inflow' is [30, 30]

# Outline

- 1 Introduction
  - Motivation
  - Contribution
- 2 Preliminaries
  - Aggregate Constraints
  - Repairs
  - Aggregate Queries
- 3 **Query Answering**
  - **Steady Aggregate Constraints**
  - **Computing Range-Consistent Answers**
  - **Experimental Results**
- 4 Conclusion and Future Work

# Steady Aggregate Constraints

- Our approach for computing consistent answers exploits a restrictions imposed on aggregate constraints

## Definition (Steady aggregate constraint)

Aggregate constraint  $\forall \vec{x} (\phi(\vec{x}) \implies \sum_{i=1}^n c_i \cdot \chi_i(\vec{y}_i) \leq K)$  is *steady* if:

- for each  $\chi_i = \langle R_i, e_i, \alpha_i \rangle$ , no measure attribute occurs in  $\alpha_i$
- measure variables occur at most once in the aggregate constraint
- no constant occurring in  $\phi$  is associated with a measure attribute

# Steady Aggregate Constraints

- Our approach for computing consistent answers exploits a restrictions imposed on aggregate constraints

## Definition (Steady aggregate constraint)

Aggregate constraint  $\forall \vec{x} (\phi(\vec{x}) \implies \sum_{i=1}^n c_i \cdot \chi_i(\vec{y}_i) \leq K)$  is *steady* if:

- 1 for each  $\chi_i = \langle R_i, e_i, \alpha_i \rangle$ , no measure attribute occurs in  $\alpha_i$
- 2 measure variables occur at most once in the aggregate constraint
- 3 no constant occurring in  $\phi$  is associated with a measure attribute

- attribute *Value* is the measure attribute of *BalanceSheets(Year, Section, Subsection, Type, Value)*

# Steady Aggregate Constraints

- Our approach for computing consistent answers exploits a restrictions imposed on aggregate constraints

## Definition (Steady aggregate constraint)

Aggregate constraint  $\forall \vec{x} (\phi(\vec{x}) \implies \sum_{i=1}^n c_i \cdot \chi_i(\vec{y}_i) \leq K)$  is *steady* if:

- 1 for each  $\chi_i = \langle R_i, e_i, \alpha_i \rangle$ , no measure attribute occurs in  $\alpha_i$
- 2 measure variables occur at most once in the aggregate constraint
- 3 no constant occurring in  $\phi$  is associated with a measure attribute

- measure variables are those variables occurring at the position of a measure attribute in  $\phi$
- $x_5$  is the measure variable for  $\phi = \text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5)$ , as it occur at the position of *Value*



# Steady Aggregate Constraints

- Our approach for computing consistent answers exploits a restrictions imposed on aggregate constraints

## Definition (Steady aggregate constraint)

Aggregate constraint  $\forall \vec{x} (\phi(\vec{x}) \implies \sum_{i=1}^n c_i \cdot \chi_i(\vec{y}_i) \leq K)$  is *steady* if:

- 1 for each  $\chi_i = \langle R_i, e_i, \alpha_i \rangle$ , no measure attribute occurs in  $\alpha_i$
  - 2 measure variables occur at most once in the aggregate constraint
  - 3 no constant occurring in  $\phi$  is associated with a measure attribute
- a constant in  $\phi$  is associated with a measure attribute if it occurs at the position of a measure attribute in  $\phi$
  - for  $\phi = \text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5)$ ,  $x_5$  cannot be a constant

# Complexity Results

- Steady aggregate constraints are expressive enough to ensure data consistency in several real-life scenarios
- The range-CQA problem is hard (even if aggregate constraints are steady)

## Theorem (Complexity of Range-CQA)

*Let  $\mathcal{D}$  be a fixed database scheme,  $\mathcal{AC}$  a fixed set of aggregate constraints on  $\mathcal{D}$ ,  $q$  a fixed aggregate query on  $\mathcal{D}$ ,  $D$  an instance of  $\mathcal{D}$ , and  $[\ell, u]$  a fixed interval.*

- Deciding whether  $CQA_{\mathcal{D}, \mathcal{AC}}^q(D) \neq \emptyset$  is NP-complete*
- Deciding whether  $CQA_{\mathcal{D}, \mathcal{AC}}^q(D) \subseteq [\ell, u]$  is  $\Delta_2^P[\log n]$ -complete*
- The lower complexity bounds still hold in the case that  $\mathcal{AC}$  is steady*

# Complexity Results

- Steady aggregate constraints are expressive enough to ensure data consistency in several real-life scenarios
- The range-CQA problem is hard (even if aggregate constraints are steady)

## Theorem (Complexity of Range-CQA)

*Let  $\mathcal{D}$  be a fixed database scheme,  $\mathcal{AC}$  a fixed set of aggregate constraints on  $\mathcal{D}$ ,  $q$  a fixed aggregate query on  $\mathcal{D}$ ,  $D$  an instance of  $\mathcal{D}$ , and  $[\ell, u]$  a fixed interval.*

- 1 *Deciding whether  $CQA_{\mathcal{D}, \mathcal{AC}}^q(D) \neq \emptyset$  is NP-complete*
- 2 *Deciding whether  $CQA_{\mathcal{D}, \mathcal{AC}}^q(D) \subseteq [\ell, u]$  is  $\Delta_2^P[\log n]$ -complete*
- 3 *The lower complexity bounds still hold in the case that  $\mathcal{AC}$  is steady*

## Basic Steps

Our approach for computing range-consistent answers w.r.t. steady aggregate constraints consists of two steps:

- 1 we compute the **cardinality of *card*-minimal repairs** by solving an ILP instance
- 2 starting from the knowledge of this cardinality, a pair of ILP instances are solved for computing the **greatest-lower bound** and the **least-upper bound** of the answers

# Steady Aggregation Expressions as Inequalities (1/2)

- A set of steady aggregate constraints  $\mathcal{AC}$  on a database scheme  $\mathcal{D}$  and an instance  $D$  of  $\mathcal{D}$  can be translated into a **set of linear inequalities**  $\mathcal{S}(\mathcal{D}, \mathcal{AC}, D)$

<i>Year</i>	<i>Section</i>	<i>Subsection</i>	<i>Type</i>	<i>Value</i>
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disburs.	payment of accounts	det	120
2008	Disburs.	capital expenditure	det	20
2008	Disburs.	long-term financing	det	80
2008	Disburs.	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

→  $z_1$ →  $z_2$ →  $z_3$ →  $z_4$ →  $z_5$ →  $z_6$ →  $z_7$ →  $z_8$ →  $z_9$ →  $z_{10}$

# Steady Aggregation Expressions as Inequalities (1/2)

- A set of steady aggregate constraints  $\mathcal{AC}$  on a database scheme  $\mathcal{D}$  and an instance  $D$  of  $\mathcal{D}$  can be translated into a **set of linear inequalities**  $\mathcal{S}(\mathcal{D}, \mathcal{AC}, D)$

<i>Year</i>	<i>Section</i>	<i>Subsection</i>	<i>Type</i>	<i>Value</i>
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disburs.	payment of accounts	det	120
2008	Disburs.	capital expenditure	det	20
2008	Disburs.	long-term financing	det	80
2008	Disburs.	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

→  $z_1$ →  $z_2$ →  $z_3$ →  $z_4$ →  $z_5$ →  $z_6$ →  $z_7$ →  $z_8$ →  $z_9$ →  $z_{10}$

## Steady Aggregation Expressions as Inequalities (1/2)

- A set of steady aggregate constraints  $\mathcal{AC}$  on a database scheme  $\mathcal{D}$  and an instance  $D$  of  $\mathcal{D}$  can be translated into a **set of linear inequalities**  $\mathcal{S}(\mathcal{D}, \mathcal{AC}, D)$

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disburs.	payment of accounts	det	120
2008	Disburs.	capital expenditure	det	20
2008	Disburs.	long-term financing	det	80
2008	Disburs.	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

→  $z_1$ →  $z_2$ →  $z_3$ →  $z_4$ →  $z_5$ →  $z_6$ →  $z_7$ →  $z_8$ →  $z_9$ →  $z_{10}$ 

$$\begin{cases} z_2 + z_3 = z_4 \\ z_5 + z_6 + z_7 = z_8 \end{cases}$$

- $\kappa_1 : \text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, x_2, \text{det}) = \chi_1(x_1, x_2, \text{aggr})$   
where  $\chi_1(x, y, z) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year} = x \wedge \text{Section} = y \wedge \text{Type} = z) \rangle$

## Steady Aggregation Expressions as Inequalities (2/2)

- Every solution of  $\mathcal{S}(\mathcal{D}, \mathcal{AC}, D)$  corresponds to a (possibly not minimal, not  $M$ -bounded) **repair** for  $D$  w.r.t.  $\mathcal{AC}$

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disburs.	payment of accounts	det	120
2008	Disburs.	capital expenditure	det	20
2008	Disburs.	long-term financing	det	80
2008	Disburs.	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

 $z_1$  $z_2$  $z_3$  $z_4$  $z_5$  $z_6$  $z_7$  $z_8$  $z_9$  $z_{10}$  $\mathcal{S}(\mathcal{D}, \{\kappa_1, \kappa_2, \kappa_3\}, D) :$ 

$$\left\{ \begin{array}{l} z_4 - z_8 = z_9 \\ z_1 + z_9 = z_{10} \\ z_2 + z_3 = z_4 \\ z_5 + z_6 + z_7 = z_8 \end{array} \right.$$



# Basic ILP

## Definition ( $ILP(\mathcal{D}, \mathcal{AC}, D)$ )

Given a database scheme  $\mathcal{D}$ , a set  $\mathcal{AC}$  of steady aggregate constraints on  $\mathcal{D}$ , and an instance  $D$  of  $\mathcal{D}$ ,  $ILP(\mathcal{D}, \mathcal{AC}, D)$  is:

$$\left\{ \begin{array}{ll} \mathbf{A} \times \vec{z} \leq \mathbf{B} & \\ z_i - M \leq 0 & -z_i - M \leq 0 \\ z_i - v_i - (M + |v_i|) \cdot \delta_i \leq 0 & -z_i + v_i - (M + |v_i|) \cdot \delta_i \leq 0 \\ z_i \in \mathbb{Z} & \delta_i \in \{0, 1\} \end{array} \right.$$

# Basic ILP

## Definition ( $ILP(\mathcal{D}, \mathcal{AC}, D)$ )

Given a database scheme  $\mathcal{D}$ , a set  $\mathcal{AC}$  of steady aggregate constraints on  $\mathcal{D}$ , and an instance  $D$  of  $\mathcal{D}$ ,  $ILP(\mathcal{D}, \mathcal{AC}, D)$  is:

$$\left\{ \begin{array}{ll} \mathbf{A} \times \vec{z} \leq \mathbf{B} & \\ z_i - M \leq 0 & -z_i - M \leq 0 \\ z_i - v_i - (M + |v_i|) \cdot \delta_i \leq 0 & -z_i + v_i - (M + |v_i|) \cdot \delta_i \leq 0 \\ z_i \in \mathbb{Z} & \delta_i \in \{0, 1\} \end{array} \right.$$

- $\mathbf{A} \times \vec{z} \leq \mathbf{B}$  is the set of inequalities  $S(\mathcal{D}, \mathcal{AC}, D)$
- $M$  bounds the absolute value of measure attributes
- $v_i$  is the database value corresponding to the variable  $z_i$

...	...	...	...	...	
2008	Receipts	beginning cash	drv	50	$\rightarrow z_1 \quad v_1 = 50$
...	...	...	...	...	

# Basic ILP

## Definition ( $ILP(\mathcal{D}, \mathcal{AC}, D)$ )

Given a database scheme  $\mathcal{D}$ , a set  $\mathcal{AC}$  of steady aggregate constraints on  $\mathcal{D}$ , and an instance  $D$  of  $\mathcal{D}$ ,  $ILP(\mathcal{D}, \mathcal{AC}, D)$  is:

$$\begin{cases} \mathbf{A} \times \vec{z} \leq \mathbf{B} \\ z_i - M \leq 0 & -z_i - M \leq 0 \\ z_i - v_i - (M + |v_i|) \cdot \delta_i \leq 0 & -z_i + v_i - (M + |v_i|) \cdot \delta_i \leq 0 \\ z_i \in \mathbb{Z} & \delta_i \in \{0, 1\} \end{cases}$$

- We defined mechanism for **counting** the number of updates:
- **if  $z_i \neq v_i$ , then  $\delta_i = 1$**
- $\sum \delta_i$  is an upper bound on the number of updates performed by the repair corresponding to the solution of  $ILP(\mathcal{D}, \mathcal{AC}, D)$

# Computing Repairs

## Theorem (Repairs )

*There is a biunique correspondence between the solutions of  $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$  and the repairs for  $D$  w.r.t  $\mathcal{AC}$ . In particular, every solution  $s$  of  $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$  corresponds to a repair  $\rho(s)$  such that the cardinality of  $\rho(s)$  is less than or equal to  $\sum \delta_i$ .*

The range-CQA is the empty interval if there is no repair

## Corollary (Empty Range-CQA)

$CQA_{\mathcal{D}, \mathcal{AC}}^q(D) = \emptyset$  iff  $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$  has no solution.

# Computing Repairs

## Theorem (Repairs )

*There is a biunique correspondence between the solutions of  $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$  and the repairs for  $D$  w.r.t  $\mathcal{AC}$ . In particular, every solution  $s$  of  $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$  corresponds to a repair  $\rho(s)$  such that the cardinality of  $\rho(s)$  is less than or equal to  $\sum \delta_i$ .*

The range-CQA is the empty interval if there is no repair

## Corollary (Empty Range-CQA)

*$CQA_{\mathcal{D}, \mathcal{AC}}^q(D) = \emptyset$  iff  $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$  has no solution.*

# Computing the Minimum Cardinality of Repairs

$$\begin{aligned} OPT(\mathcal{D}, \mathcal{AC}, D) &:= \\ & \text{minimize } \sum_i \delta_i \text{ subject to} \\ & ILP(\mathcal{D}, \mathcal{AC}, D) \end{aligned}$$

## Corollary (Cardinality of *Card*-minimal repairs)

*The optimal value of  $OPT(\mathcal{D}, \mathcal{AC}, D)$  coincides with the cardinality of any card-minimal repair for  $D$  w.r.t.  $\mathcal{AC}$ .*

- The solution of  $OPT(\mathcal{D}, \mathcal{AC}, D)$  is exploited to compute (not empty) range-consistent answers

# SUM-queries (1/2)

- Let  $\lambda$  be the cardinality of any *card*-minimal repair. The solutions of

$$\begin{cases} \text{ILP}(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \end{cases}$$

one-to-one correspond to *card-minimal repairs* for  $D$  w.r.t.  $\mathcal{AC}$

- For  $q = \text{SELECT SUM}(A) \text{ FROM } R \text{ WHERE } \alpha$  we define  $\mathcal{T}(q)$  as

$$\sum_{t: t \in R \wedge t \models \alpha} z_{t,A},$$

i.e., the sum of variables  $z$  associated with tuples of  $R$  satisfying the WHERE condition

- minimizing (resp. maximizing)  $\mathcal{T}(q)$  subject to  $\begin{cases} \text{ILP}(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \end{cases}$  result in the *minimum (resp. maximum) value of  $q$*  on all the databases resulting from applying *card*-minimal repairs

# SUM-queries (1/2)

- Let  $\lambda$  be the cardinality of any *card*-minimal repair. The solutions of

$$\begin{cases} \text{ILP}(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \end{cases}$$

one-to-one correspond to *card-minimal repairs* for  $D$  w.r.t.  $\mathcal{AC}$

- For  $q = \text{SELECT SUM}(\mathbf{A}) \text{ FROM } R \text{ WHERE } \alpha$  we define  $\mathcal{T}(q)$  as

$$\sum_{t: t \in R \wedge t \models \alpha} z_{t, \mathbf{A}},$$

i.e., the sum of variables  $z$  associated with tuples of  $R$  satisfying the WHERE condition

- minimizing (resp. maximizing)  $\mathcal{T}(q)$  subject to  $\begin{cases} \text{ILP}(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \end{cases}$  result in the *minimum (resp. maximum) value of  $q$*  on all the databases resulting from applying *card*-minimal repairs



## SUM-queries (1/2)

- Let  $\lambda$  be the cardinality of any *card*-minimal repair. The solutions of

$$\begin{cases} \text{ILP}(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \end{cases}$$

one-to-one correspond to *card-minimal repairs* for  $D$  w.r.t.  $\mathcal{AC}$

- For  $q = \text{SELECT SUM}(\mathbf{A}) \text{ FROM } R \text{ WHERE } \alpha$  we define  $\mathcal{T}(q)$  as

$$\sum_{t: t \in R \wedge t \models \alpha} z_{t, \mathbf{A}},$$

i.e., the sum of variables  $z$  associated with tuples of  $R$  satisfying the `WHERE` condition

- minimizing (resp. maximizing)  $\mathcal{T}(q)$  subject to  $\begin{cases} \text{ILP}(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \end{cases}$  result in the *minimum (resp. maximum) value of  $q$*  on all the databases resulting from applying *card*-minimal repairs

# SUM-queries (2/2)

greatest-lower bound

$$\begin{aligned} OPT_{glb}^{SUM}(\mathcal{D}, \mathcal{AC}, q, D) := \\ \text{minimize } T(q) \text{ subject to} \\ \begin{cases} ILP(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \end{cases} \end{aligned}$$

least-upper bound

$$\begin{aligned} OPT_{lub}^{SUM}(\mathcal{D}, \mathcal{AC}, q, D) := \\ \text{maximize } T(q) \text{ subject to} \\ \begin{cases} ILP(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \end{cases} \end{aligned}$$

## Theorem (Range-Consistent Answer of SUM-query)

For a SUM-query  $q$ , either  $CQA_{\mathcal{D}, \mathcal{AC}}^q(D) = \emptyset$ , or  $CQA_{\mathcal{D}, \mathcal{AC}}^q(D) = [\ell, u]$ , where

- 1  $\ell$  is the value returned by  $OPT_{glb}^{SUM}(\mathcal{D}, \mathcal{AC}, q, D)$
- 2  $u$  the value returned by  $OPT_{lub}^{SUM}(\mathcal{D}, \mathcal{AC}, q, D)$ .

## MAX-queries (1/2)

- Additional inequalities are exploited to encode the **MAX function**
- Let  $\mathcal{I}(q)$  be the set of indexes of variables  $z$  associated with the tuples selected by MAX-query  $q$ , we define  $ln(q)$  as

$$\left\{ \begin{array}{ll} z_j - z_i - 2M \cdot \mu_i \leq 0 & \forall j, i \in \mathcal{I}(q), j \neq i \\ \sum_{i \in \mathcal{I}(q)} \mu_i = |\mathcal{I}(q)| - 1 & \\ x_i - M \cdot \mu_i \leq 0; & -x_i - M \cdot \mu_i \leq 0; \\ z_i - x_i - 2M \cdot (1 - \mu_i) \leq 0; & -z_i + x_i - 2M \cdot (1 - \mu_i) \leq 0; \\ x_i - M \leq 0; & -x_i - M \leq 0; \\ x_i \in \mathbb{Z}; & \mu_i \in \{0, 1\}; \quad \forall i \in \mathcal{I}(q); \end{array} \right.$$

- $z_i - x_i = \begin{cases} z_i & \text{if } z_i \text{ takes the maximum value among variables } z_j \\ 0 & \text{otherwise} \end{cases}$

## MAX-queries (1/2)

- Additional inequalities are exploited to encode the **MAX function**
- Let  $\mathcal{I}(q)$  be the set of indexes of variables  $z$  associated with the tuples selected by MAX-query  $q$ , we define  $ln(q)$  as

$$\left\{ \begin{array}{ll} z_j - z_i - 2M \cdot \mu_i \leq 0 & \forall j, i \in \mathcal{I}(q), j \neq i \\ \sum_{i \in \mathcal{I}(q)} \mu_i = |\mathcal{I}(q)| - 1 & \\ x_i - M \cdot \mu_i \leq 0; & -x_i - M \cdot \mu_i \leq 0; \\ z_i - x_i - 2M \cdot (1 - \mu_i) \leq 0; & -z_i + x_i - 2M \cdot (1 - \mu_i) \leq 0; \\ x_i - M \leq 0; & -x_i - M \leq 0; \\ x_i \in \mathbb{Z}; & \mu_i \in \{0, 1\}; \quad \forall i \in \mathcal{I}(q); \end{array} \right.$$

- $z_i - x_i = \begin{cases} z_i & \text{if } z_i \text{ takes the maximum value among variables } z_j \\ 0 & \text{otherwise} \end{cases}$

## MAX-queries (1/2)

- Additional inequalities are exploited to encode the **MAX function**
- Let  $\mathcal{I}(q)$  be the set of indexes of variables  $z$  associated with the tuples selected by MAX-query  $q$ , we define  $ln(q)$  as

$$\left\{ \begin{array}{ll} z_j - z_i - 2M \cdot \mu_i \leq 0 & \forall j, i \in \mathcal{I}(q), j \neq i \\ \sum_{i \in \mathcal{I}(q)} \mu_i = |\mathcal{I}(q)| - 1 & \\ x_i - M \cdot \mu_i \leq 0; & -x_i - M \cdot \mu_i \leq 0; \\ z_i - x_i - 2M \cdot (1 - \mu_i) \leq 0; & -z_i + x_i - 2M \cdot (1 - \mu_i) \leq 0; \\ x_i - M \leq 0; & -x_i - M \leq 0; \\ x_i \in \mathbb{Z}; & \mu_i \in \{0, 1\}; \quad \forall i \in \mathcal{I}(q); \end{array} \right.$$

- $z_i - x_i = \begin{cases} z_i & \text{if } z_i \text{ takes the } \mathbf{maximum} \text{ value among variables } z_j \\ 0 & \text{otherwise} \end{cases}$

## MAX-queries (2/2)

$$\begin{aligned} OPT_{glb}^{MAX}(\mathcal{D}, \mathcal{AC}, q, D) := \\ \text{minimize } \sum_{i \in \mathcal{I}(q)} (z_i - x_i) \\ \text{subject to} \\ \left\{ \begin{array}{l} \mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \\ \ln(q) \end{array} \right. \end{aligned}$$

$$\begin{aligned} OPT_{lub}^{MAX}(\mathcal{D}, \mathcal{AC}, q, D) := \\ \text{maximize } \sum_{i \in \mathcal{I}(q)} (z_i - x_i) \\ \text{subject to} \\ \left\{ \begin{array}{l} \mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \\ \ln(q) \end{array} \right. \end{aligned}$$

### Theorem (Range-Consistent Answer of MAX-query)

For a MAX-query  $q$ , either  $CQA_{\mathcal{D}, \mathcal{AC}}^q(D) = \emptyset$ , or  $CQA_{\mathcal{D}, \mathcal{AC}}^q(D) = [\ell, u]$

- 1  $\ell$  is the value returned by  $OPT_{glb}^{MAX}(\mathcal{D}, \mathcal{AC}, q, D)$
- 2  $u$  is the value returned by  $OPT_{lub}^{MAX}(\mathcal{D}, \mathcal{AC}, q, D)$ .

- A similar (symmetric) result holds for **MIN-queries**

## MAX-queries (2/2)

$$OPT_{glb}^{MAX}(\mathcal{D}, \mathcal{AC}, q, D) :=$$

*minimize*  $\sum_{i \in \mathcal{I}(q)} (z_i - x_i)$

*subject to*

$$\begin{cases} \mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \\ \ln(q) \end{cases}$$

$$OPT_{lub}^{MAX}(\mathcal{D}, \mathcal{AC}, q, D) :=$$

*maximize*  $\sum_{i \in \mathcal{I}(q)} (z_i - x_i)$

*subject to*

$$\begin{cases} \mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \\ \ln(q) \end{cases}$$

### Theorem (Range-Consistent Answer of MAX-query)

For a MAX-query  $q$ , either  $CQA_{\mathcal{D}, \mathcal{AC}}^q(D) = \emptyset$ , or  $CQA_{\mathcal{D}, \mathcal{AC}}^q(D) = [\ell, u]$

- 1  $\ell$  is the value returned by  $OPT_{glb}^{MAX}(\mathcal{D}, \mathcal{AC}, q, D)$
- 2  $u$  is the value returned by  $OPT_{lub}^{MAX}(\mathcal{D}, \mathcal{AC}, q, D)$ .

- A similar (symmetric) result holds for MIN-queries

## MAX-queries (2/2)

$$OPT_{glb}^{MAX}(\mathcal{D}, \mathcal{AC}, q, D) :=$$

*minimize*  $\sum_{i \in \mathcal{I}(q)} (z_i - x_i)$

*subject to*

$$\begin{cases} \mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \\ \ln(q) \end{cases}$$

$$OPT_{lub}^{MAX}(\mathcal{D}, \mathcal{AC}, q, D) :=$$

*maximize*  $\sum_{i \in \mathcal{I}(q)} (z_i - x_i)$

*subject to*

$$\begin{cases} \mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D) \\ \lambda = \sum \delta_i \\ \ln(q) \end{cases}$$

### Theorem (Range-Consistent Answer of MAX-query)

For a MAX-query  $q$ , either  $CQA_{\mathcal{D}, \mathcal{AC}}^q(D) = \emptyset$ , or  $CQA_{\mathcal{D}, \mathcal{AC}}^q(D) = [\ell, u]$

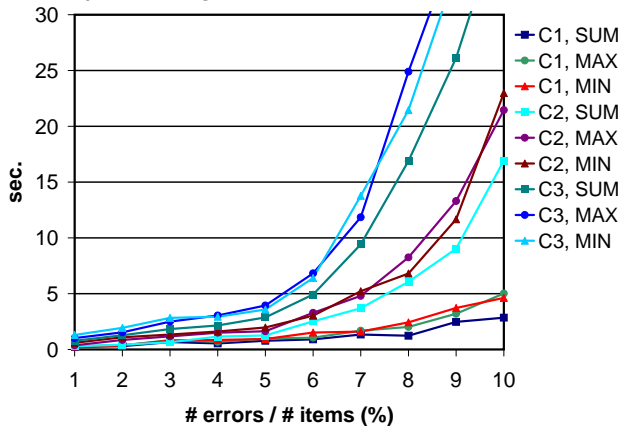
- 1  $\ell$  is the value returned by  $OPT_{glb}^{MAX}(\mathcal{D}, \mathcal{AC}, q, D)$
- 2  $u$  is the value returned by  $OPT_{lub}^{MAX}(\mathcal{D}, \mathcal{AC}, q, D)$ .

- A similar (symmetric) result holds for **MIN-queries**



## Experiment 1 on data set *Balance Sheets*

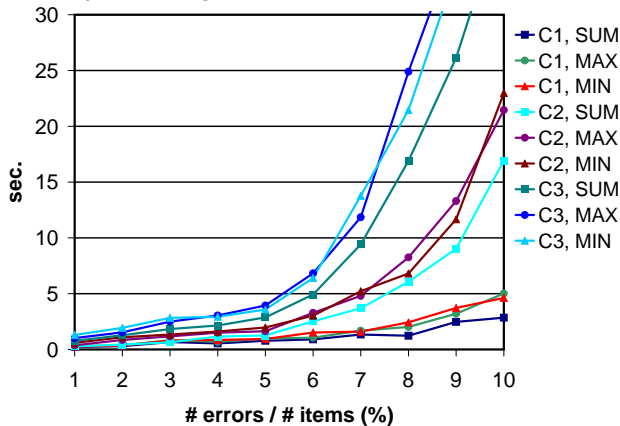
- Average time needed for computing range-consistent answers vs. the percentage of erroneous values



- 3 years balance sheets of companies C1, C2, C3 containing 346, 780, and 1234 tuples, respectively
- typically the percentage of errors is less than 5% of acquired data

## Experiment 1 on data set *Balance Sheets*

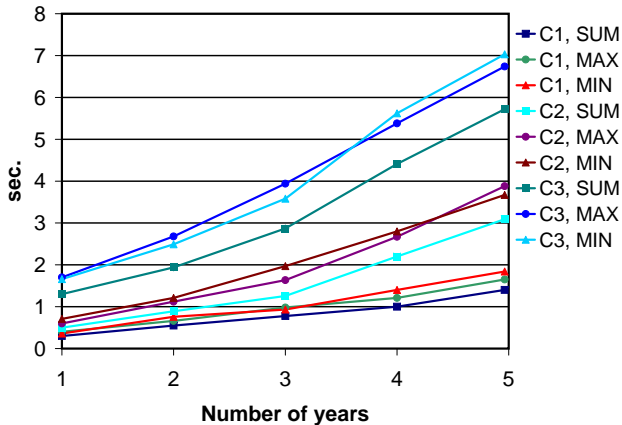
- Average time needed for computing range-consistent answers vs. the percentage of erroneous values



- 3 years balance sheets of companies C1, C2, C3 containing 346, 780, and 1234 tuples, respectively
- typically the percentage of errors is less than 5% of acquired data

## Experiment 2 on data set *Balance Sheets*

- An insight on the impact of the database size on the performance of our technique (5% of erroneous values)



- every 1-year balance sheet of companies C1, C2, C3 contains about 115, 260, and 410 tuples, respectively

# Outline

## 1 Introduction

- Motivation
- Contribution

## 2 Preliminaries

- Aggregate Constraints
- Repairs
- Aggregate Queries

## 3 Query Answering

- Steady Aggregate Constraints
- Computing Range-Consistent Answers
- Experimental Results

## 4 Conclusion and Future Work

## Conclusion and ...

- We have introduced a framework for computing range-consistent answers of  $\text{MAX-}$ ,  $\text{MIN-}$ , and  $\text{SUM-}$ queries in numerical databases violating a given set of aggregate constraints
- Our approach exploits a transformation into integer linear programming (ILP), thus allowing us to exploit well-known techniques for solving ILP problems
- Experimental results prove the feasibility of the proposed approach in real-life application scenarios

## ... Future Work

Further work will be devoted to

- devising strategies for computing range-consistent answers of other form of queries (e.g. *AVG*, *GROUPBY* clause,...)
- devising strategies for improving performance of our technique (e.g., reducing the number of variables and inequalities used)
- devising a transformation for non-steady constraints (and queries with `WHERE` clause containing also measure attributes)
- remove the assumption that measure attributes are bounded in value (range-consistent answers can be  $\pm\infty$ )

Thank you!

... any question?

## Related Work

- The *range-consistent query answer* semantics was introduced in [Arenas et Al (TCS 2003)], as a more specific notion of consistent answer w.r.t. the original definition of [Arenas et Al (PODS 1999)] for dealing with aggregate queries (in the presence of FDs)
- Range-CQAs were further investigated in [Fuxman et Al (SIGMOD 2005)] for aggregate queries with grouping under key constraints
- [Flesca et Al (TODS 2010)] investigated several problems regarding the extraction of reliable information from data violating aggregate constraints (including CQA for atomic ground queries)
- None of these works investigated range-CQAs to aggregate queries under of aggregate constraints.



## Related Work

- The *range-consistent query answer* semantics was introduced in [Arenas et Al (TCS 2003)], as a more specific notion of consistent answer w.r.t. the original definition of [Arenas et Al (PODS 1999)] for dealing with aggregate queries (in the presence of FDs)
- Range-CQAs were further investigated in [Fuxman et Al (SIGMOD 2005)] for aggregate queries with grouping under key constraints
- [Flesca et Al (TODS 2010)] investigated several problems regarding the extraction of reliable information from data violating aggregate constraints (including CQA for atomic ground queries)
- None of these works investigated range-CQAa to aggregate queries under of aggregate constraints.





## Related Work

- The *range-consistent query answer* semantics was introduced in [Arenas et Al (TCS 2003)], as a more specific notion of consistent answer w.r.t. the original definition of [Arenas et Al (PODS 1999)] for dealing with aggregate queries (in the presence of FDs)
- Range-CQAs were further investigated in [Fuxman et Al (SIGMOD 2005)] for aggregate queries with grouping under key constraints
- [Flesca et Al (TODS 2010)] investigated several problems regarding the extraction of reliable information from data violating aggregate constraints (including CQA for atomic ground queries)
- None of these works investigated range-CQAa to aggregate queries under of aggregate constraints.

## Related Work

- The *range-consistent query answer* semantics was introduced in [Arenas et Al (TCS 2003)], as a more specific notion of consistent answer w.r.t. the original definition of [Arenas et Al (PODS 1999)] for dealing with aggregate queries (in the presence of FDs)
- Range-CQAs were further investigated in [Fuxman et Al (SIGMOD 2005)] for aggregate queries with grouping under key constraints
- [Flesca et Al (TODS 2010)] investigated several problems regarding the extraction of reliable information from data violating aggregate constraints (including CQA for atomic ground queries)
- None of these works investigated range-CQAa to aggregate queries under of aggregate constraints.

## Related Work

-  Arenas, M., Bertossi, L.E., Chomicki, J.: *Consistent query answers in inconsistent databases*. In: Proc. 18<sup>th</sup> ACM Symp. on Principles of Database Systems (PODS). (1999) 68–79
-  Arenas, M., Bertossi, L.E., Chomicki, J., He, X., Raghavan, V., Spinrad, J.: *Scalar aggregation in inconsistent databases*. Theor. Comput. Sci. (TCS) Vol. 3(296) (2003) 405–434
-  Fuxman, A., Fazli, E., Miller, R.J.: *Conquer: Efficient management of inconsistent databases*. In: Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD). (2005) 155–166
-  Flesca, S., Furfaro, F., Parisi, F.: *Querying and Repairing Inconsistent Numerical Databases*. ACM Transactions on Database Systems (TODS), Vol 35 (2), 2010

# Semantics of Aggregate Constraints

- An aggregate constraint is an aggregation expression that a database should satisfy
- The database  $D$  satisfies the aggregate constraint

$$\kappa : \forall \vec{X} (\phi(\vec{X}) \implies \sum_{i=1}^n c_i \cdot \chi_i(\vec{y}_i) \leq K)$$

if, for all the substitutions of the variables in  $\vec{X}$  with constants making the conjunction of atoms on the  $LHS(\kappa)$  true, the inequality on the  $RHS(\kappa)$  holds on  $D$ .

- A database  $D$  is consistent w.r.t. a set of aggregate constraints  $\mathcal{AC}$  if  $D \models \mathcal{AC}$

# Semantics of Aggregate Constraints

- An aggregate constraint is an aggregation expression that a database should satisfy
- The database  $D$  satisfies the aggregate constraint

$$\kappa : \forall \vec{X} (\phi(\vec{X}) \implies \sum_{i=1}^n c_i \cdot \chi_i(\vec{y}_i) \leq K)$$

if, for all the substitutions of the variables in  $\vec{X}$  with constants making the conjunction of atoms on the  $LHS(\kappa)$  *true*, the inequality on the  $RHS(\kappa)$  holds on  $D$ .

- A database  $D$  is consistent w.r.t. a set of aggregate constraints  $\mathcal{AC}$  if  $D \models \mathcal{AC}$

# Semantics of Aggregate Constraints

- An aggregate constraint is an aggregation expression that a database should satisfy
- The database  $D$  satisfies the aggregate constraint

$$\kappa : \forall \vec{X} (\phi(\vec{X}) \implies \sum_{i=1}^n c_i \cdot \chi_i(\vec{y}_i) \leq K)$$

if, for all the substitutions of the variables in  $\vec{X}$  with constants making the conjunction of atoms on the  $LHS(\kappa)$  *true*, the inequality on the  $RHS(\kappa)$  holds on  $D$ .

- A database  $D$  is consistent w.r.t. a set of aggregate constraints  $\mathcal{AC}$  if  $D \models \mathcal{AC}$

# Example of Aggregate Constraint (1/3)

*BalanceSheets*

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_1$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

- $\chi_1(x, y) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year} = x \wedge \text{Subsection} = y) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) - (\chi_1(x_1, \text{'total cash receipts'}) - \chi_1(x_1, \text{'total disbursements'})) = 0$



# Example of Aggregate Constraint (1/3)

*BalanceSheets*

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_1$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

- $\chi_1(x, y) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year} = x \wedge \text{Subsection} = y) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) - (\chi_1(x_1, \text{'total cash receipts'}) - \chi_1(x_1, \text{'total disbursements'})) = 0$

# Example of Aggregate Constraint (1/3)

*BalanceSheets*

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_1$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

- $\chi_1(x, y) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year} = x \wedge \text{Subsection} = y) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) - (\chi_1(x_1, \text{'total cash receipts'}) - \chi_1(x_1, \text{'total disbursements'})) = 0$

# Example of Aggregate Constraint (1/3)

*BalanceSheets*

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_1$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

- $\chi_1(x, y) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year} = x \wedge \text{Subsection} = y) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) - (\chi_1(x_1, \text{'total cash receipts'}) - \chi_1(x_1, \text{'total disbursements'})) = 0$

## Example of Aggregate Constraint (2/3)

*BalanceSheets*

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_2$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

- $\chi_1(x, y) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year} = x \wedge \text{Subsection} = y) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'ending cash balance'}) - (\chi_1(x_1, \text{'beginning cash'}) + \chi_1(x_1, \text{'net cash inflow'})) = 0$

## Example of Aggregate Constraint (2/3)

*BalanceSheets*

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_2$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

- $\chi_1(x, y) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year} = x \wedge \text{Subsection} = y) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'ending cash balance'}) - (\chi_1(x_1, \text{'beginning cash'}) + \chi_1(x_1, \text{'net cash inflow'})) = 0$

## Example of Aggregate Constraint (2/3)

*BalanceSheets*

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_2$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

- $\chi_1(x, y) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year} = x \wedge \text{Subsection} = y) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'ending cash balance'}) - (\chi_1(x_1, \text{'beginning cash'}) + \chi_1(x_1, \text{'net cash inflow'})) = 0$

# Example of Aggregate Constraint (2/3)

*BalanceSheets*

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_2$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

- $\chi_1(x, y) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year} = x \wedge \text{Subsection} = y) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'ending cash balance'}) - (\chi_1(x_1, \text{'beginning cash'}) + \chi_1(x_1, \text{'net cash inflow'})) = 0$

# Example of Aggregate Constraint (3/3)

## BalanceSheets

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_3$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year

- $\chi_2(x, y, z) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year}=x \wedge \text{Section}=y \wedge \text{Type}=z) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_2(x_1, x_2, \text{'det'}) = \chi_2(x_1, x_2, \text{'aggr'})$



# Example of Aggregate Constraint (3/3)

## BalanceSheets

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_3$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year

- $\chi_2(x, y, z) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year}=x \wedge \text{Section}=y \wedge \text{Type}=z) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_2(x_1, x_2, \text{'det'}) = \chi_2(x_1, x_2, \text{'aggr'})$

# Example of Aggregate Constraint (3/3)

## BalanceSheets

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_3$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year

- $\chi_2(x, y, z) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year}=x \wedge \text{Section}=y \wedge \text{Type}=z) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_2(x_1, x_2, \text{'det'}) = \chi_2(x_1, x_2, \text{'aggr'})$

# Example of Aggregate Constraint (3/3)

*BalanceSheets*

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\kappa_3$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year

- $\chi_2(x, y, z) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year}=x \wedge \text{Section}=y \wedge \text{Type}=z) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_2(x_1, x_2, \text{'det'}) = \chi_2(x_1, x_2, \text{'aggr'})$

## Two examples of *card*-minimal repairs

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$                        $\rho_2$   
 $\rightarrow 130$                        $\rightarrow 150$

- $\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year
- $\kappa_2$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- $\kappa_3$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*

Two examples of *card*-minimal repairs

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$                        $\rho_2$   
 $\rightarrow 130$                        $\rightarrow 150$

- $\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year
- $\kappa_2$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- $\kappa_3$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*

Two examples of *card*-minimal repairs

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$        $\rho_2$   
 $\rightarrow 130$        $\rightarrow 150$

- $\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year
- $\kappa_2$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- $\kappa_3$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*

Two examples of *card*-minimal repairs

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$                        $\rho_2$   
 $\rightarrow 130$                        $\rightarrow 150$

- $\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year
- $\kappa_2$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- $\kappa_3$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*

Two examples of *card*-minimal repairs

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$                        $\rho_2$   
 $\rightarrow$  130                       $\rightarrow$  150

- $\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year
- $\kappa_2$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- $\kappa_3$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*



Two examples of *card*-minimal repairs

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$                        $\rho_2$   
 $\rightarrow 130$                        $\rightarrow 150$

- $\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year
- $\kappa_2$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- $\kappa_3$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*

Two examples of *card*-minimal repairs

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$                        $\rho_2$   
 $\rightarrow 130$                        $\rightarrow 150$

- $\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year
- $\kappa_2$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- $\kappa_3$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*

Two examples of *card*-minimal repairs

Year	Section	Subsection	Type	Value
2008	Receipts	beginning cash	drv	50
2008	Receipts	cash sales	det	100
2008	Receipts	receivables	det	120
2008	Receipts	total cash receipts	aggr	250
2008	Disbursements	payment of accounts	det	120
2008	Disbursements	capital expenditure	det	20
2008	Disbursements	long-term financing	det	80
2008	Disbursements	total disbursements	aggr	220
2008	Balance	net cash inflow	drv	30
2008	Balance	ending cash balance	drv	80

$\rho_1$                        $\rho_2$   
 $\rightarrow 130$                        $\rightarrow 150$

- $\kappa_1$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year
- $\kappa_2$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- $\kappa_3$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*

## Repairing non-numerical data (1/2)

- We assume that inconsistencies involve numerical attributes (measure attributes) only
- Non-measure attributes are assumed to be consistent
- In many real-life situations, even if integrity violations of measure data can coexist with integrity violations involving non-measure data, these inconsistencies can be fixed separately

## Repairing non-numerical data (1/2)

- We assume that inconsistencies involve numerical attributes (measure attributes) only
- Non-measure attributes are assumed to be consistent
- In many real-life situations, even if integrity violations of measure data can coexist with integrity violations involving non-measure data, these inconsistencies can be fixed separately

# Computing the Minimum Cardinality of Repairs - Example

- For the *BalanceSheets* database where  $\mathcal{AC} = \{\kappa_1, \kappa_2, \kappa_3\}$ ,  $OPT(\mathcal{D}, \mathcal{AC}, D)$  is

minimize  $\sum_i \delta_i$  subject to

$$\left\{ \begin{array}{l} z_4 - z_8 = z_9 \\ z_1 + z_9 = z_{10} \\ z_2 + z_3 = z_4 \\ z_5 + z_6 + z_7 = z_8 \\ z_i - M \leq 0 \\ -z_i - M \leq 0 \\ z_i \in \mathbb{Z} \\ \delta_i \in \{0, 1\} \end{array} \right. \quad \begin{array}{l} z_1 - 50 - (M + 50) \cdot \delta_1 \leq 0 \\ z_2 - 100 - (M + 100) \cdot \delta_2 \leq 0 \\ z_3 - 120 - (M + 120) \cdot \delta_3 \leq 0 \\ z_4 - 250 - (M + 250) \cdot \delta_4 \leq 0 \\ z_5 - 120 - (M + 120) \cdot \delta_5 \leq 0 \\ z_6 - 20 - (M + 20) \cdot \delta_6 \leq 0 \\ z_7 - 80 - (M + 80) \cdot \delta_7 \leq 0 \\ z_8 - 220 - (M + 220) \cdot \delta_8 \leq 0 \\ z_9 - 30 - (M + 30) \cdot \delta_9 \leq 0 \\ z_{10} - 80 - (M + 80) \cdot \delta_{10} \leq 0 \end{array} \quad \begin{array}{l} -z_1 + 50 - (M + 50) \cdot \delta_1 \leq 0 \\ -z_2 + 100 - (M + 100) \cdot \delta_2 \leq 0 \\ -z_3 + 120 - (M + 120) \cdot \delta_3 \leq 0 \\ -z_4 + 250 - (M + 250) \cdot \delta_4 \leq 0 \\ -z_5 + 120 - (M + 120) \cdot \delta_5 \leq 0 \\ -z_6 + 20 - (M + 20) \cdot \delta_6 \leq 0 \\ -z_7 + 80 - (M + 80) \cdot \delta_7 \leq 0 \\ -z_8 + 220 - (M + 220) \cdot \delta_8 \leq 0 \\ -z_9 + 30 - (M + 30) \cdot \delta_9 \leq 0 \\ -z_{10} + 80 - (M + 80) \cdot \delta_{10} \leq 0 \end{array}$$

- encoding of the aggregate constraints
- bounds on measure values
- mechanism for counting the number of updates

## Repairing non-numerical data (2/2)

- In the balance sheet scenario, errors in the OCR-mediated acquisition of non-measure attributes (such as lacks of correspondences between real and acquired strings denoting item descriptions) can be repaired in a pre-processing step using a dictionary, by searching for the strings in the dictionary which are the most similar to the acquired ones
- [Fazzinga, et Al (IIDB 2006)] described a system adopting such a dictionary-based repairing strategy for string attributes

## Repairing non-numerical data (2/2)

- In the balance sheet scenario, errors in the OCR-mediated acquisition of non-measure attributes (such as lacks of correspondences between real and acquired strings denoting item descriptions) can be repaired in a pre-processing step using a dictionary, by searching for the strings in the dictionary which are the most similar to the acquired ones
- [Fazzinga, et Al (IIDB 2006)] described a system adopting such a dictionary-based repairing strategy for string attributes



## Example of non-steady aggregate constraint

- Consider the relation scheme  $R_2(\underline{\text{Project}}, \text{Department}, \text{Costs})$  database scheme
- and the following constraint: *There is at most one “expensive” project* (a project is considered expensive if its costs are not less than 20K)
- This constraint can be expressed by the following aggregate constraint:  $\chi() \leq 1$ , where  $\chi = \langle R_2, 1, (\text{Costs} \geq 20K) \rangle$
- As attribute *Costs* is a measure attribute of  $R_2$ , and it occurs in the formula  $\alpha$  of the aggregation function  $\chi$ , the above-introduced aggregate constraint is not steady (condition (1) of the Definition of steady aggregate constraint is not satisfied).

## Example of non-steady aggregate constraint

- Consider the relation scheme  $R_2(\underline{Project}, Department, Costs)$  database scheme
- and the following constraint: *There is at most one “expensive” project* (a project is considered expensive if its costs are not less than 20K)
- This constraint can be expressed by the following aggregate constraint:  $\chi() \leq 1$ , where  $\chi = \langle R_2, 1, (Costs \geq 20K) \rangle$
- As attribute *Costs* is a measure attribute of  $R_2$ , and it occurs in the formula  $\alpha$  of the aggregation function  $\chi$ , the above-introduced aggregate constraint is not steady (condition (1) of the Definition of steady aggregate constraint is not satisfied).

## Example of non-steady aggregate constraint

- Consider the relation scheme  $R_2(\underline{Project}, Department, Costs)$  database scheme
- and the following constraint: *There is at most one "expensive" project* (a project is considered expensive if its costs are not less than 20K)
- This constraint can be expressed by the following aggregate constraint:  $\chi() \leq 1$ , where  $\chi = \langle R_2, 1, (Costs \geq 20K) \rangle$
- As attribute *Costs* is a measure attribute of  $R_2$ , and it occurs in the formula  $\alpha$  of the aggregation function  $\chi$ , the above-introduced aggregate constraint is not steady (condition (1) of the Definition of steady aggregate constraint is not satisfied).

## Example of non-steady aggregate constraint

- Consider the relation scheme  $R_2(\underline{Project}, Department, Costs)$  database scheme
- and the following constraint: *There is at most one "expensive" project* (a project is considered expensive if its costs are not less than 20K)
- This constraint can be expressed by the following aggregate constraint:  $\chi() \leq 1$ , where  $\chi = \langle R_2, 1, (Costs \geq 20K) \rangle$
- As attribute *Costs* is a measure attribute of  $R_2$ , and it occurs in the formula  $\alpha$  of the aggregation function  $\chi$ , the above-introduced aggregate constraint is not steady (condition (1) of the Definition of steady aggregate constraint is not satisfied).

# Experiment Setting

- We experimentally validated our framework for computing range-CQAs on data set *Balance Sheets* containing real-life balance-sheet data
- We used LINDO API 4.0 as ILP solver, and a PC with Intel Pentium 4 Processor at 3.00 GHz and 4GB RAM

# Constraints and Queries of Experiments on data set Balance Sheets (1/3)

- We considered the aggregate constraints  $\mathcal{AC} = \{\kappa_1, \kappa_2, \kappa_3\}$

$\kappa_1$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

- $\chi_1(x, y) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year}=x \wedge \text{Subsection}=y) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) - (\chi_1(x_1, \text{'total cash receipts'}) - \chi_1(x_1, \text{'total disbursements'})) = 0$

$\kappa_2$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'ending cash balance'}) - (\chi_1(x_1, \text{'beginning cash'}) + \chi_1(x_1, \text{'net cash inflow'})) = 0$

# Constraints and Queries of Experiments on data set Balance Sheets (1/3)

- We considered the aggregate constraints  $\mathcal{AC} = \{\kappa_1, \kappa_2, \kappa_3\}$

$\kappa_1$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

- $\chi_1(x, y) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year} = x \wedge \text{Subsection} = y) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) - (\chi_1(x_1, \text{'total cash receipts'}) - \chi_1(x_1, \text{'total disbursements'})) = 0$

$\kappa_2$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'ending cash balance'}) - (\chi_1(x_1, \text{'beginning cash'}) + \chi_1(x_1, \text{'net cash inflow'})) = 0$

# Constraints and Queries of Experiments on data set Balance Sheets (1/3)

- We considered the aggregate constraints  $\mathcal{AC} = \{\kappa_1, \kappa_2, \kappa_3\}$

$\kappa_1$  for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

- $\chi_1(x, y) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year} = x \wedge \text{Subsection} = y) \rangle$
- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) - (\chi_1(x_1, \text{'total cash receipts'}) - \chi_1(x_1, \text{'total disbursements'})) = 0$

$\kappa_2$  for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

- $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'ending cash balance'}) - (\chi_1(x_1, \text{'beginning cash'}) + \chi_1(x_1, \text{'net cash inflow'})) = 0$



# Constraints and Queries of Experiments on data set Balance Sheets (2/3)

- $\kappa_3$  for each section and year, the sum of the values of all *detail* items must be equal to the value of the *aggregate* item of the same section and year
- $\chi_2(x, y, z) = \langle \text{BalanceSheets}, \text{Value}, (\text{Year}=x \wedge \text{Section}=y \wedge \text{Type}=z) \rangle$
  - $\text{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_2(x_1, x_2, \text{'det'}) = \chi_2(x_1, x_2, \text{'aggr'})$

# Constraints and Queries of Experiments on data set Balance Sheets (3/3)

- We considered queries  $q_1, q_2, q_3$  obtained from the following “template” query by replacing  $f$  with MAX, MIN, SUM, respectively:

```
SELECT  $f$ (Value)  
FROM BalanceSheets  
WHERE Subection = ‘cash sales’
```




- along with the queries  $q_4, q_5, q_6$  obtained from the following template by replacing  $f$  with MAX, MIN, SUM, respectively:

```
SELECT  $f$ (Value)  
FROM BalanceSheets  
WHERE Section = ‘Receipts’  $\wedge$  Type  $\neq$  ‘aggr’
```

# Complexity Classes

- *P*TIME: the class of decision problems solvable in polynomial time by deterministic Turing Machines; this class is also denoted as *P*;
- *NP*: the class of decision problems solvable in polynomial time by nondeterministic Turing Machines;
- $\Delta_2^P$ : the class of decision problems solvable in polynomial time by deterministic Turing machines with an *NP* oracle; this class is also denoted as  $P^{NP}$ ;
- $\Delta_2^P[\log(n)]$ : the class of decision problems solvable in polynomial time by deterministic Turing machines with an *NP* oracle which is invoked  $\mathcal{O}(\log(n))$  times; this class is also denoted as  $P^{NP[\log(n)]}$ ;

# For Further Reading

-  Arenas, M., Bertossi, L.E., Chomicki, J.:  
Consistent query answers in inconsistent databases.  
In: Proc. 18<sup>th</sup> ACM Symp. on Principles of Database Systems  
(PODS). (1999) 68–79
-  Flesca, S., Furfaro, F., Parisi, F.:  
Querying and Repairing Inconsistent Numerical Databases.  
ACM Transactions on Database Systems (TODS), Vol 35 (2), 2010
-  Fazzinga, B., Flesca, S., Furfaro, F., Parisi, F.:  
Dart: A data acquisition and repairing tool.  
In: Proc. Int. Workshop on Incons. and Incompl. in Databases  
(IIDB). (2006) 297–317