Consistent Answers to Boolean Aggregate Queries under Aggregate Constraints

Sergio Flesca, Filippo Furfaro, Francesco Parisi

DEIS
University of Calabria
87036 Rende (CS), Italy

21^{st} *DEXA Conference*

Bilbao, Spain

30 August - 3 September 2010

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

- Digitized balance sheets can be analyzed for determining financial reliability of companies
- Examples of queries which can support this kind of analysis are:
  - q_1 : for each year, is the value of *net cash inflow* greater than a given threshold?
  - q_2 : for years 2008 and 2009, is the sum of *receivables* greater than *payment of accounts*?
- The mere evaluation of these queries on inconsistent data may yield a wrong picture of the real world
- To support any analysis task, it is mandatory to retrieve "reliable" information even if the data are inconsistent

Querying Inconsistent Data

- Digitized balance sheets can be analyzed for determining financial reliability of companies
- Examples of queries which can support this kind of analysis are:
  - q_1 : for each year, is the value of *net cash inflow* greater than a given threshold?
  - q_2 : for years 2008 and 2009, is the sum of *receivables* greater than *payment of accounts*?
- The mere evaluation of these queries on inconsistent data may yield a wrong picture of the real world
- To support any analysis task, it is mandatory to retrieve "reliable" information even if the data are inconsistent

Querying Inconsistent Data

- Digitized balance sheets can be analyzed for determining financial reliability of companies
- Examples of queries which can support this kind of analysis are:
  - q_1 : for each year, is the value of *net cash inflow* greater than a given threshold?
  - q_2 : for years 2008 and 2009, is the sum of *receivables* greater than *payment of accounts*?
- The mere evaluation of these queries on inconsistent data may yield a wrong picture of the real world
- To support any analysis task, it is mandatory to retrieve “reliable” information even if the data are inconsistent

Querying Inconsistent Data

- Digitized balance sheets can be analyzed for determining financial reliability of companies
- Examples of queries which can support this kind of analysis are:
  - q_1 : for each year, is the value of *net cash inflow* greater than a given threshold?
  - q_2 : for years 2008 and 2009, is the sum of *receivables* greater than *payment of accounts*?
- The mere evaluation of these queries on inconsistent data may yield a wrong picture of the real world
- To support any analysis task, it is mandatory to retrieve “reliable” information even if the data are inconsistent

Consistent Query Answer (CQA)

- A great deal of attention has been recently devoted the problem of extracting reliable information from data violating a given set of integrity constraints
- Most of the approaches are based on the notions of *repairs* and *consistent query answer* (CQA) introduced in [Arenas et Al (PODS 1999)].
  - A repair is a database resulting from fixing the original database in a minimal way (preserving information of the original database as much as possible).
  - Consistent answers are those that can be obtained from every possible repair of the database

Consistent Query Answer (CQA)

- A great deal of attention has been recently devoted the problem of extracting reliable information from data violating a given set of integrity constraints
- Most of the approaches are based on the notions of *repairs* and *consistent query answer* (CQA) introduced in [Arenas et Al (PODS 1999)].
  - A repair is a database resulting from fixing the original database in a minimal way (preserving information of the original database as much as possible).
  - Consistent answers are those that can be obtained from every possible repair of the database

Consistent Query Answer (CQA)

- A great deal of attention has been recently devoted the problem of extracting reliable information from data violating a given set of integrity constraints
- Most of the approaches are based on the notions of *repairs* and *consistent query answer* (CQA) introduced in [Arenas et Al (PODS 1999)].
  - A repair is a database resulting from fixing the original database in a minimal way (preserving information of the original database as much as possible).
  - Consistent answers are those that can be obtained from every possible repair of the database

Consistent Query Answer (CQA)

- A great deal of attention has been recently devoted the problem of extracting reliable information from data violating a given set of integrity constraints
- Most of the approaches are based on the notions of *repairs* and *consistent query answer* (CQA) introduced in [Arenas et Al (PODS 1999)].
  - A repair is a database resulting from fixing the original database in a minimal way (preserving information of the original database as much as possible).
  - Consistent answers are those that can be obtained from every possible repair of the database

Aggregate Constraints

- 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 must the equal to the total cash receipts*
- Aggregate constraints allow us to define algebraic relations between aggregate values extracted from the database
- In [Flesca et Al (TODS 2010)] the CQA problem for atomic queries in the presence of aggregate constraints was investigated

Aggregate Constraints

- 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 must the equal to the total cash receipts*
- Aggregate constraints allow us to define algebraic relations between aggregate values extracted from the database
- In [Flesca et Al (TODS 2010)] the CQA problem for atomic queries in the presence of aggregate constraints was investigated

Aggregate Queries

- Atomic queries are not expressive enough
  - analysis tasks, such as those performed on balance sheet data, cannot be supported by simple atomic queries
- We study the CQA of *boolean aggregate queries*
- This kind of queries allow us to express conditions consisting of linear inequalities on aggregate-sum functions

q_1 : for each year, is the value of *net cash inflow* greater than a given threshold?

q_2 : for years 2008 and 2009, is the sum of *receivables* greater than *payment of accounts*?

Aggregate Queries

- Atomic queries are not expressive enough
  - analysis tasks, such as those performed on balance sheet data, cannot be supported by simple atomic queries
- We study the CQA of *boolean aggregate queries*
- This kind of queries allow us to express conditions consisting of linear inequalities on aggregate-sum functions

q_1 : for each year, is the value of *net cash inflow* greater than a given threshold?

q_2 : for years 2008 and 2009, is the sum of *receivables* greater than *payment of accounts*?

CQA for Aggregate Queries under Aggregate Constraints

- We characterized the computational complexity of the CQA problem for boolean aggregate queries in the presence of aggregate constraints
- We devised a strategy for computing consistent answers to boolean aggregate queries in the presence of aggregate constraints
- Our approach computes consistent answers by solving Integer Linear Programming (ILP) problem instances
- Our approach enables the computation of CQA by means of well-known techniques for solving ILP problems
- We experimentally validated our approach

CQA for Aggregate Queries under Aggregate Constraints

- We characterized the computational complexity of the CQA problem for boolean aggregate queries in the presence of aggregate constraints
- We devised a strategy for computing consistent answers to boolean aggregate queries in the presence of aggregate constraints
- Our approach computes consistent answers by solving Integer Linear Programming (ILP) problem instances
- Our approach enables the computation of CQA by means of well-known techniques for solving ILP problems
- We experimentally validated our approach

CQA for Aggregate Queries under Aggregate Constraints

- We characterized the computational complexity of the CQA problem for boolean aggregate queries in the presence of aggregate constraints
- We devised a strategy for computing consistent answers to boolean aggregate queries in the presence of aggregate constraints
- Our approach computes consistent answers by solving Integer Linear Programming (ILP) problem instances
- Our approach enables the computation of CQA by means of well-known techniques for solving ILP problems
- We experimentally validated our approach

CQA for Aggregate Queries under Aggregate Constraints

- We characterized the computational complexity of the CQA problem for boolean aggregate queries in the presence of aggregate constraints
- We devised a strategy for computing consistent answers to boolean aggregate queries in the presence of aggregate constraints
- Our approach computes consistent answers by solving Integer Linear Programming (ILP) problem instances
- Our approach enables the computation of CQA by means of well-known techniques for solving ILP problems
- We experimentally validated our approach

CQA for Aggregate Queries under Aggregate Constraints

- We characterized the computational complexity of the CQA problem for boolean aggregate queries in the presence of aggregate constraints
- We devised a strategy for computing consistent answers to boolean aggregate queries in the presence of aggregate constraints
- Our approach computes consistent answers by solving Integer Linear Programming (ILP) problem instances
- Our approach enables the computation of CQA by means of well-known techniques for solving ILP problems
- We experimentally validated our approach

Outline

Aggregation Expressions

- Both aggregate constraints and aggregate queries will be expressed by *aggregation expressions*

Definition (Aggregation Expression)

An aggregation expression is of the form:

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

- $\phi(\vec{x})$ is a conjunction of relation atoms
- $c_1, \ldots, c_n, K$ are constants
- each $\chi_i(\vec{y}_i)$ is an aggregation function (with *variables*$(\vec{y}_i) \subseteq \vec{x}$)

- 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

Aggregation Expressions

- Both aggregate constraints and aggregate queries will be expressed by *aggregation expressions*

Definition (Aggregation Expression)

An aggregation expression is of the form:

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

- $\phi(\vec{x})$ is a conjunction of relation atoms
- $c_1, \ldots, c_n, K$ are constants
- each $\chi_i(\vec{y}_i)$ is an aggregation function (with $variables(\vec{y}_i) \subseteq \vec{x}$)

- 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

Aggregation Expressions

- Both aggregate constraints and aggregate queries will be expressed by *aggregation expressions*

Definition (Aggregation Expression)

An aggregation expression is of the form:

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

- $\phi(\vec{x})$ is a conjunction of relation atoms
- $c_1, \ldots, c_n, K$ are constants
- each $\chi_i(\vec{y}_i)$ is an aggregation function (with *variables*$(\vec{y}_i) \subseteq \vec{x}$)

- 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

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

κ_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

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

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

κ_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

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

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

κ_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

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

Repairing strategy

- A repair for a database w.r.t. a set of aggregate constraints is a set of value updates making the database consistent
- We adopt the strategy proposed in [Flesca et Al (TODS 2010)] for repairing data inconsistent w.r.t. a set of aggregate constraints
- 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

- A repair for a database w.r.t. a set of aggregate constraints is a set of value updates making the database consistent
- We adopt the strategy proposed in [Flesca et Al (TODS 2010)] for repairing data inconsistent w.r.t. a set of aggregate constraints
- 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

- A repair for a database w.r.t. a set of aggregate constraints is a set of value updates making the database consistent
- We adopt the strategy proposed in [Flesca et Al (TODS 2010)] for repairing data inconsistent w.r.t. a set of aggregate constraints
- 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

ρ_1	ρ_2
$\longrightarrow$ 130	
	$\longrightarrow$ 150

κ_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

κ_2 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

κ_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***	ρ_1	ρ_2
2008	Receipts	beginning cash	drv	50		
2008	Receipts	cash sales	det	100	$\longrightarrow$ 130	
2008	Receipts	receivables	det	120		$\longrightarrow$ 150
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		

- κ_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
- κ_2 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- κ_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***	ρ_1	ρ_2
2008	Receipts	beginning cash	drv	50		
2008	Receipts	cash sales	det	100	$\longrightarrow$ 130	
2008	Receipts	receivables	det	120		$\longrightarrow$ 150
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		

- κ_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
- κ_2 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- κ_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***	ρ_1	ρ_2
2008	Receipts	beginning cash	drv	50		
2008	Receipts	cash sales	det	100	$\longrightarrow$ **130**	
2008	Receipts	receivables	det	120		$\longrightarrow$ **150**
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		

- κ_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
- κ_2 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- κ_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 and Consistent Answers

- Aggregate queries are defined by aggregation expressions
- For *BalanceSheets*(*Year*, *Section*, *Subsection*, *Type*, *Value*)

q_1 : for each year, is the value of *net cash inflow* greater than 20?

- $\chi_2(x, y) = \langle \mathit{BalanceSheets}, \mathit{Value}, (\mathit{Year}{=}x \wedge \mathit{Subsection}{=}y) \rangle$
- $\mathit{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) \geq 20$

Definition (Consistent query answer)

Let $\mathcal{D}$ be a database scheme, D an instance of $\mathcal{D}$, $\mathcal{AC}$ a set of aggregate constraints on $\mathcal{D}$ and q an aggregate query over $\mathcal{D}$. The *consistent query answer* to q on D w.r.t. $\mathcal{AC}$ is true iff, for each *card*-minimal repair ρ for D w.r.t. $\mathcal{AC}$, it holds that q evaluates to *true* on the database resulting from performing all the updates in ρ.

Aggregate Queries and Consistent Answers

- Aggregate queries are defined by aggregation expressions
- For *BalanceSheets*(*Year*, *Section*, *Subsection*, *Type*, *Value*)

q_1 : for each year, is the value of *net cash inflow* greater than 20?

- $\chi_2(x, y) = \langle BalanceSheets,\ Value,\ (Year{=}x \wedge Subsection{=}y)\rangle$
- $BalanceSheets(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) \geq 20$

Definition (Consistent query answer)

Let $\mathcal{D}$ be a database scheme, D an instance of $\mathcal{D}$, $\mathcal{AC}$ a set of aggregate constraints on $\mathcal{D}$ and q an aggregate query over $\mathcal{D}$. The *consistent query answer* to q on D w.r.t. $\mathcal{AC}$ is true iff, for each *card*-minimal repair ρ for D w.r.t. $\mathcal{AC}$, it holds that q evaluates to *true* on the database resulting from performing all the updates in ρ.

Aggregate Queries and Consistent Answers

- Aggregate queries are defined by aggregation expressions
- For *BalanceSheets*(*Year*, *Section*, *Subsection*, *Type*, *Value*)

q_1 : for each year, is the value of *net cash inflow* greater than 20?

- $\chi_2(x, y) = \langle BalanceSheets,\ Value,\ (Year{=}x \wedge Subsection{=}y\ )\rangle$
- $BalanceSheets(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) \geq 20$

Definition (Consistent query answer)

Let $\mathcal{D}$ be a database scheme, D an instance of $\mathcal{D}$, $\mathcal{AC}$ a set of aggregate constraints on $\mathcal{D}$ and q an aggregate query over $\mathcal{D}$. The *consistent query answer* to q on D w.r.t. $\mathcal{AC}$ is true iff, for each *card*-minimal repair ρ for D w.r.t. $\mathcal{AC}$, it holds that q evaluates to *true* on the database resulting from performing all the updates in ρ.

Outline

Steady Aggregate Constraints and Queries

- Our approach for computing consistent answers exploits a restrictions imposed on aggregation expressions

Definition (Steady aggregation expression)

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

1. for each $\chi_i = \langle R_i, e_i, \alpha_i \rangle$, no measure attribute occurs in α_i
2. measure variables occur at most once in the aggregation expression
3. no constant occurring in ϕ is associated with a measure attribute

Steady Aggregate Constraints and Queries

- Our approach for computing consistent answers exploits a restrictions imposed on aggregation expressions

Definition (Steady aggregation expression)

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

1. for each $\chi_i = \langle R_i, e_i, \alpha_i \rangle$, no measure attribute occurs in α_i
2. measure variables occur at most once in the aggregation expression
3. no constant occurring in ϕ is associated with a measure attribute

- measure attributes are those that can be updated by a repair
- attribute *Value* is the measure attribute of *BalanceSheets*(*Year*, *Section*, *Subsection*, *Type*, *Value*)

Steady Aggregate Constraints and Queries

- Our approach for computing consistent answers exploits a restrictions imposed on aggregation expressions

Definition (Steady aggregation expression)

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

1. for each $\chi_i = \langle R_i, e_i, \alpha_i \rangle$, no measure attribute occurs in α_i
2. measure variables occur at most once in the aggregation expression
3. no constant occurring in ϕ is associated with a measure attribute

- measure variables are those variables occurring at the position of a measure attribute in ϕ
- x_5 is the measure variable for $\phi = BalanceSheets(x_1, x_2, x_3, x_4, x_5)$

Steady Aggregate Constraints and Queries

- Our approach for computing consistent answers exploits a restrictions imposed on aggregation expressions

Definition (Steady aggregation expression)

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

1. for each $\chi_i = \langle R_i, e_i, \alpha_i \rangle$, no measure attribute occurs in α_i
2. measure variables occur at most once in the aggregation expression
3. no constant occurring in ϕ is associated with a measure attribute

- a constant in ϕ is associated with a measure attribute if it occurs at the position of a measure attribute in ϕ
- for $\phi = BalanceSheets(x_1, x_2, x_3, x_4, x_5)$, x_5 cannot be a constant

Complexity Results

- Steady aggregation expressions are less expressive than (general) aggregation expressions
- The loss in expressiveness is not dramatic, as steady aggregate constraints/queries can model several real-life conditions
- The consistent query answer problem is hard also when both the aggregate constraints and the query are steady

Theorem (Complexity of 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 over D, and D an instance of $\mathcal{D}$. Deciding whether $\text{CQA}_{\mathcal{D},\mathcal{AC},q}(D)$ is true is $\Delta_2^p[\log n]$-complete, even if both $\mathcal{AC}$ and q are steady.

Complexity Results

- Steady aggregation expressions are less expressive than (general) aggregation expressions
- The loss in expressiveness is not dramatic, as steady aggregate constraints/queries can model several real-life conditions
- The consistent query answer problem is hard also when both the aggregate constraints and the query are steady

Theorem (Complexity of 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 over D, and D an instance of $\mathcal{D}$. Deciding whether $\textsc{cqa}_{\mathcal{D},\mathcal{AC},q}(D)$ is true is $\Delta_2^p[\log n]$-complete, even if both $\mathcal{AC}$ and q are steady.

Basic Steps

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

1. first, we compute the cardinality of *card*-minimal repairs by solving an ILP instance
2. starting from the knowledge of this cardinality another ILP instance is solved for computing CQA

Steady Aggregation Expressions as Inequalities (1/2)

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

Year	*Section*	*Subsection*	*Type*	*Value*	
2008	Receipts	beginning cash	drv	50	$\rightarrow z_1$
2008	Receipts	cash sales	det	100	$\rightarrow z_2$
2008	Receipts	receivables	det	120	$\rightarrow z_3$
2008	Receipts	total cash receipts	aggr	250	$\rightarrow z_4$
2008	Disburs.	payment of accounts	det	120	$\rightarrow z_5$
2008	Disburs.	capital expenditure	det	20	$\rightarrow z_6$
2008	Disburs.	long-term financing	det	80	$\rightarrow z_7$
2008	Disburs.	total disbursements	aggr	220	$\rightarrow z_8$
2008	Balance	net cash inflow	drv	30	$\rightarrow z_9$
2008	Balance	ending cash balance	drv	80	$\rightarrow z_{10}$

Steady Aggregation Expressions as Inequalities (1/2)

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

Year	***Section***	***Subsection***	***Type***	***Value***	
2008	Receipts	beginning cash	drv	50	$\rightarrow z_1$
2008	Receipts	cash sales	det	100	$\rightarrow z_2$
2008	Receipts	receivables	det	120	$\rightarrow z_3$
2008	Receipts	total cash receipts	aggr	250	$\rightarrow z_4$
2008	Disburs.	payment of accounts	det	120	$\rightarrow z_5$
2008	Disburs.	capital expenditure	det	20	$\rightarrow z_6$
2008	Disburs.	long-term financing	det	80	$\rightarrow z_7$
2008	Disburs.	total disbursements	aggr	220	$\rightarrow z_8$
2008	Balance	net cash inflow	drv	30	$\rightarrow z_9$
2008	Balance	ending cash balance	drv	80	$\rightarrow z_{10}$

Steady Aggregation Expressions as Inequalities (1/2)

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

Year	*Section*	*Subsection*	*Type*	*Value*	
2008	Receipts	beginning cash	drv	50	$\rightarrow z_1$
2008	Receipts	cash sales	det	100	$\rightarrow z_2$
2008	Receipts	receivables	det	120	$\rightarrow z_3$
2008	Receipts	total cash receipts	aggr	250	$\rightarrow z_4$
2008	Disburs.	payment of accounts	det	120	$\rightarrow z_5$
2008	Disburs.	capital expenditure	det	20	$\rightarrow z_6$
2008	Disburs.	long-term financing	det	80	$\rightarrow z_7$
2008	Disburs.	total disbursements	aggr	220	$\rightarrow z_8$
2008	Balance	net cash inflow	drv	30	$\rightarrow z_9$
2008	Balance	ending cash balance	drv	80	$\rightarrow z_{10}$

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

- κ_1 : $BalanceSheets(x_1, x_2, x_3, x_4, x_5) \Longrightarrow \chi_2(x_1, x_2, \text{det}) = \chi_2(x_1, x_2, \text{aggr})$
  where $\chi_1(x, y, z) = \langle BalanceSheets,\ Value,\ (Year = x \wedge Section = y \wedge Type = z) \rangle$

Steady Aggregation Expressions as Inequalities (1/2)

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

Year	*Section*	*Subsection*	*Type*	*Value*	
2008	Receipts	beginning cash	drv	50	$\rightarrow z_1$
2008	Receipts	cash sales	det	100	$\rightarrow z_2$
2008	Receipts	receivables	det	120	$\rightarrow z_3$
2008	Receipts	total cash receipts	aggr	250	$\rightarrow z_4$
2008	Disburs.	payment of accounts	det	120	$\rightarrow z_5$
2008	Disburs.	capital expenditure	det	20	$\rightarrow z_6$
2008	Disburs.	long-term financing	det	80	$\rightarrow z_7$
2008	Disburs.	total disbursements	aggr	220	$\rightarrow z_8$
2008	Balance	net cash inflow	drv	30	$\rightarrow z_9$
2008	Balance	ending cash balance	drv	80	$\rightarrow z_{10}$

$z_9 \geq 20$

- q_1 : $BalanceSheets(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) \geq 20$ where $\chi_2(x, y) = \langle BalanceSheets, Value, (Year{=}x \wedge Subsection{=}y) \rangle$

Steady Aggregation Expressions as Inequalities (2/2)

- Every solution of $\mathcal{S}(\mathcal{D}, \mathcal{E}, D)$ corresponds to a database update U such that the database resulting from applying U to D satisfies the aggregation expressions $\mathcal{E}$
- For $\mathcal{AC} = \{\kappa_1, \kappa_2, \kappa_3\}$, every solution of $\mathcal{S}(\mathcal{D}, \mathcal{AC}, D)$ corresponds to a (possibly not minimal) repair for D w.r.t. $\mathcal{AC}$

Year	Section	Subsection	Type	Value	
2008	Receipts	beginning cash	drv	50	z_1
2008	Receipts	cash sales	det	100	z_2
2008	Receipts	receivables	det	120	z_3
2008	Receipts	total cash receipts	aggr	250	z_4
2008	Disburs.	payment of accounts	det	120	z_5
2008	Disburs.	capital expenditure	det	20	z_6
2008	Disburs.	long-term financing	det	80	z_7
2008	Disburs.	total disbursements	aggr	220	z_8
2008	Balance	net cash inflow	drv	30	z_9
2008	Balance	ending cash balance	drv	80	z_{10}

$$\mathcal{S}(\mathcal{D}, \mathcal{AC}, D): \begin{cases} z_4 - z_8 = z_9 \\ z_1 + z_9 = z_{10} \\ z_2 + z_3 = z_4 \\ z_5 + z_6 + z_7 = z_8 \end{cases}$$

Steady Aggregation Expressions as Inequalities (2/2)

- Every solution of $\mathcal{S}(\mathcal{D}, \mathcal{E}, D)$ corresponds to a database update U such that the database resulting from applying U to D satisfies the aggregation expressions $\mathcal{E}$
- For $\mathcal{AC} = \{\kappa_1, \kappa_2, \kappa_3\}$, every solution of $\mathcal{S}(\mathcal{D}, \mathcal{AC}, D)$ corresponds to a (possibly not minimal) repair for D w.r.t. $\mathcal{AC}$

Year	***Section***	***Subsection***	***Type***	***Value***	
2008	Receipts	beginning cash	drv	50	z_1
2008	Receipts	cash sales	det	100	z_2
2008	Receipts	receivables	det	120	z_3
2008	Receipts	total cash receipts	aggr	250	z_4
2008	Disburs.	payment of accounts	det	120	z_5
2008	Disburs.	capital expenditure	det	20	z_6
2008	Disburs.	long-term financing	det	80	z_7
2008	Disburs.	total disbursements	aggr	220	z_8
2008	Balance	net cash inflow	drv	30	z_9
2008	Balance	ending cash balance	drv	80	z_{10}

$$\mathcal{S}(\mathcal{D}, \mathcal{AC}, D): \begin{cases} z_4 - z_8 = z_9 \\ z_1 + z_9 = z_{10} \\ z_2 + z_3 = z_4 \\ z_5 + z_6 + z_7 = z_8 \end{cases}$$

Computing the Minimum Cardinality of Repairs (1/2)

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

Let $\mathcal{S}(\mathcal{D}, \mathcal{AC}, D)$ be $\mathbf{A} \times \vec{z} \leq \mathbf{B}$.
$\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$ is an ILP of the form:

$$\begin{cases} \mathbf{A} \times \vec{z} \leq \mathbf{B}; & \\ w_i = z_i - v_i & \\ z_i - M \leq 0; & -z_i - M \leq 0; \\ w_i - M\delta_i \leq 0; & -w_i - M\delta_i \leq 0; \\ z_i, w_i \in \mathbb{Z}; & \delta_i \in \{0, 1\}; \end{cases}$$

Computing the Minimum Cardinality of Repairs (1/2)

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

Let $\mathcal{S}(\mathcal{D}, \mathcal{AC}, D)$ be $\mathbf{A} \times \vec{z} \leq \mathbf{B}$.
$\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$ is an ILP of the form:

$$\begin{cases} \mathbf{A} \times \vec{z} \leq \mathbf{B}; & \\ w_i = z_i - v_i & \\ z_i - M \leq 0; & -z_i - M \leq 0; \\ w_i - M\delta_i \leq 0; & -w_i - M\delta_i \leq 0; \\ z_i, w_i \in \mathbb{Z}; & \delta_i \in \{0, 1\}; \end{cases}$$

- v_i is the database value corresponding to the variable z_i

...	...	...	...	...
2008	Receipts	beginning cash	drv	50
...	...	...	...	...

$\rightarrow z_1$ $\quad v_1 = 50$

- w_i and δ_i are new variables

Computing the Minimum Cardinality of Repairs (1/2)

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

Let $\mathcal{S}(\mathcal{D}, \mathcal{AC}, D)$ be $\mathbf{A} \times \vec{z} \leq \mathbf{B}$.
$\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$ is an ILP
of the form:

$$\begin{cases} \mathbf{A} \times \vec{z} \leq \mathbf{B}; & \\ w_i = z_i - v_i & \\ z_i - M \leq 0; & -z_i - M \leq 0; \\ w_i - M\delta_i \leq 0; & -w_i - M\delta_i \leq 0; \\ z_i, w_i \in \mathbb{Z}; & \delta_i \in \{0, 1\}; \end{cases}$$

The constant M is introduced for a twofold objective:

1. considering solutions of $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$ which correspond to polynomial-size repairs for D w.r.t. $\mathcal{AC}$

Computing the Minimum Cardinality of Repairs (1/2)

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

Let $\mathcal{S}(\mathcal{D}, \mathcal{AC}, D)$ be $\mathbf{A} \times \vec{z} \leq \mathbf{B}$. $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$ is an ILP of the form:

$$\begin{cases} \mathbf{A} \times \vec{z} \leq \mathbf{B}; & \\ w_i = z_i - v_i & \\ z_i - M \leq 0; & -z_i - M \leq 0; \\ w_i - M\delta_i \leq 0; & -w_i - M\delta_i \leq 0; \\ z_i, w_i \in \mathbb{Z}; & \delta_i \in \{0, 1\}; \end{cases}$$

The constant M is introduced for a twofold objective:

1. considering solutions of $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$ which correspond to polynomial-size repairs for D w.r.t. $\mathcal{AC}$
2. building a mechanism for counting the number of updates (i.e., the number of variables w_i which are assigned a value different from 0)

Computing the Minimum Cardinality of Repairs (1/2)

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

Let $\mathcal{S}(\mathcal{D}, \mathcal{AC}, D)$ be $\mathbf{A} \times \vec{z} \leq \mathbf{B}$. $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$ is an ILP of the form:

$$\begin{cases} \mathbf{A} \times \vec{z} \leq \mathbf{B}; & \\ w_i = z_i - v_i & \\ z_i - M \leq 0; & -z_i - M \leq 0; \\ w_i - M\delta_i \leq 0; & -w_i - M\delta_i \leq 0; \\ z_i, w_i \in \mathbb{Z}; & \delta_i \in \{0, 1\}; \end{cases}$$

- The value of M derives from a well-known general result shown in [Papadimitriou (JACM 1981)] regarding the existence of bounded solutions of systems of linear equalities
- The sum of the values assigned to variables δ_i is an upper bound on the number of updates performed by the repair corresponding to the solution of $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$

Computing the Minimum Cardinality of Repairs (2/2)

Theorem (Cardinality of *Card*-minimal repairs)

Let $\mathcal{D}$ be a database scheme, $\mathcal{AC}$ a set of steady aggregate constraints on $\mathcal{D}$, and D an instance of $\mathcal{D}$. A repair for D w.r.t. $\mathcal{AC}$ exists iff $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$ has at least one solution, and the optimal value of the optimization problem:

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

coincides with the cardinality of any card-minimal repair for D w.r.t. $\mathcal{AC}$.

- The solution of $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ is exploited to compute consistent query answers

Computing the Minimum Cardinality of Repairs - Example

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

$$\textit{minimize } \sum_i \delta_i \textit{ subject to}$$

$$\begin{cases}
z_4 - z_8 = z_9 & w_3 = z_3 - 120 & w_{10} = z_{10} - 80 \\
z_1 + z_9 = z_{10} & w_4 = z_4 - 250 & w_i - M\delta_i \leq 0 \\
z_2 + z_3 = z_4 & w_5 = z_5 - 120 & -w_i - M\delta_i \leq 0 \\
z_5 + z_6 + z_7 = z_8 & w_6 = z_6 - 20 & z_i - M \leq 0 \\
w_1 = z_1 - 50 & w_7 = z_7 - 80 & -z_i - M \leq 0 \\
w_2 = z_2 - 100 & w_8 = z_8 - 220 & \delta_i \in \{0, 1\} \\
 & w_9 = z_9 - 30 &
\end{cases}$$

- encoding of the aggregate constraints
- mechanism for counting the number of updates and considering polynomial solution w.r.t. the size of the database

Computing Consistent Answers (1/2)

- Given an aggregate query q, consider $\mathcal{ILP}(\mathcal{D}, \mathcal{AC} \cup \{\neg q\}, D)$
- $\mathcal{ILP}(\mathcal{D}, \mathcal{AC} \cup \{\neg q\}, D)$ is obtained by treating the aggregation expression corresponding to the negation of q as a constraint
  - For the *BalanceSheets* database with $\mathcal{AC} = \{\kappa_1, \kappa_2, \kappa_3\}$ and $q_1 : BalanceSheets(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) \geq 20$

$\mathcal{ILP}(\mathcal{D}, \{\kappa_1, \kappa_2, \kappa_3\} \cup \{, \neg q_1\}, D)$:

$$\begin{cases} z_4 - z_8 = z_9 & w_3 = z_3 - 120 & w_{10} = z_{10} - 80 \\ z_1 + z_9 = z_{10} & w_4 = z_4 - 250 & w_i - M\delta_i \leq 0 \\ z_2 + z_3 = z_4 & w_5 = z_5 - 120 & -w_i - M\delta_i \leq 0 \\ z_5 + z_6 + z_7 = z_8 & w_6 = z_6 - 20 & z_i - M \leq 0 \\ z_9 \leq 20 & w_7 = z_7 - 80 & -z_i - M \leq 0 \\ w_1 = z_1 - 50 & w_8 = z_8 - 220 & z_i, w_i \in \mathbb{Z} \\ w_2 = z_2 - 100 & w_9 = z_9 - 30 & \delta_i \in \{0, 1\} \end{cases}$$

- encoding of the negated aggregate query

Computing Consistent Answers (1/2)

- Given an aggregate query q, consider $\mathcal{ILP}(\mathcal{D}, \mathcal{AC} \cup \{\neg q\}, D)$
- $\mathcal{ILP}(\mathcal{D}, \mathcal{AC} \cup \{\neg q\}, D)$ is obtained by treating the aggregation expression corresponding to the negation of q as a constraint
  - For the *BalanceSheets* database with $\mathcal{AC} = \{\kappa_1, \kappa_2, \kappa_3\}$ and q_1 : $BalanceSheets(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) \geq 20$

$\mathcal{ILP}(\mathcal{D}, \{\kappa_1, \kappa_2, \kappa_3\} \cup \{, \neg q_1\}, D)$:

$$\begin{cases} z_4 - z_8 = z_9 & w_3 = z_3 - 120 & w_{10} = z_{10} - 80 \\ z_1 + z_9 = z_{10} & w_4 = z_4 - 250 & w_i - M\delta_i \leq 0 \\ z_2 + z_3 = z_4 & w_5 = z_5 - 120 & -w_i - M\delta_i \leq 0 \\ z_5 + z_6 + z_7 = z_8 & w_6 = z_6 - 20 & z_i - M \leq 0 \\ z_9 \leq 20 & w_7 = z_7 - 80 & -z_i - M \leq 0 \\ w_1 = z_1 - 50 & w_8 = z_8 - 220 & z_i, w_i \in \mathbb{Z} \\ w_2 = z_2 - 100 & w_9 = z_9 - 30 & \delta_i \in \{0, 1\} \end{cases}$$

- encoding of the negated aggregate query

Computing Consistent Answers (1/2)

- Given an aggregate query q, consider $\mathcal{ILP}(\mathcal{D}, \mathcal{AC} \cup \{\neg q\}, D)$
- $\mathcal{ILP}(\mathcal{D}, \mathcal{AC} \cup \{\neg q\}, D)$ is obtained by treating the aggregation expression corresponding to the negation of q as a constraint
  - For the *BalanceSheets* database with $\mathcal{AC} = \{\kappa_1, \kappa_2, \kappa_3\}$ and q_1 : $BalanceSheets(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) \geq 20$

$\mathcal{ILP}(\mathcal{D}, \{\kappa_1, \kappa_2, \kappa_3\} \cup \{, \neg q_1\}, D)$:

$$\begin{cases} z_4 - z_8 = z_9 & w_3 = z_3 - 120 & w_{10} = z_{10} - 80 \\ z_1 + z_9 = z_{10} & w_4 = z_4 - 250 & w_i - M\delta_i \leq 0 \\ z_2 + z_3 = z_4 & w_5 = z_5 - 120 & -w_i - M\delta_i \leq 0 \\ z_5 + z_6 + z_7 = z_8 & w_6 = z_6 - 20 & z_i - M \leq 0 \\ z_9 \leq 20 & w_7 = z_7 - 80 & -z_i - M \leq 0 \\ w_1 = z_1 - 50 & w_8 = z_8 - 220 & z_i, w_i \in \mathbb{Z} \\ w_2 = z_2 - 100 & w_9 = z_9 - 30 & \delta_i \in \{0, 1\} \end{cases}$$

- encoding of the negated aggregate query

Computing Consistent Answers (2/2)

- The solutions of $\mathcal{ILP}(\mathcal{D}, \mathcal{AC} \cup \{\neg q\}, D)$ correspond to (possibly non-minimal) repairs for D w.r.t. $\mathcal{AC}$ such that q evaluates to *false* on the repaired databases
- Let $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D) = \begin{cases} \mathcal{ILP}(\mathcal{D}, \mathcal{AC} \cup \{\neg q\}, D) \\ \lambda = \sum_i \delta_i \end{cases}$
where λ is the value returned by $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$.

Theorem (Consistent Query Answer)

Let $\mathcal{D}$ be a database scheme, $\mathcal{AC}$ a set of steady aggregate constraints on $\mathcal{D}$, q a steady aggregate query on $\mathcal{D}$, and D an instance of $\mathcal{D}$. The consistent query answer to q over D w.r.t. $\mathcal{AC}$ is true iff $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$ has no solution.

Computing Consistent Answers (2/2)

- The solutions of $\mathcal{ILP}(\mathcal{D}, \mathcal{AC} \cup \{\neg q\}, D)$ correspond to (possibly non-minimal) repairs for D w.r.t. $\mathcal{AC}$ such that q evaluates to *false* on the repaired databases
- Let $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D) = \begin{cases} \mathcal{ILP}(\mathcal{D}, \mathcal{AC} \cup \{\neg q\}, D) \\ \lambda = \sum_i \delta_i \end{cases}$
  where λ is the value returned by $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$.

Theorem (Consistent Query Answer)

Let $\mathcal{D}$ be a database scheme, $\mathcal{AC}$ a set of steady aggregate constraints on $\mathcal{D}$, q a steady aggregate query on $\mathcal{D}$, and D an instance of $\mathcal{D}$. The consistent query answer to q over D w.r.t. $\mathcal{AC}$ is true iff $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$ has no solution.

Computing Consistent Answers (2/2)

- The solutions of $\mathcal{ILP}(\mathcal{D}, \mathcal{AC} \cup \{\neg q\}, D)$ correspond to (possibly non-minimal) repairs for D w.r.t. $\mathcal{AC}$ such that q evaluates to *false* on the repaired databases
- Let $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D) = \begin{cases} \mathcal{ILP}(\mathcal{D}, \mathcal{AC} \cup \{\neg q\}, D) \\ \lambda = \sum_i \delta_i \end{cases}$
  where λ is the value returned by $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$.

Theorem (Consistent Query Answer)

Let $\mathcal{D}$ be a database scheme, $\mathcal{AC}$ a set of steady aggregate constraints on $\mathcal{D}$, q a steady aggregate query on $\mathcal{D}$, and D an instance of $\mathcal{D}$. The consistent query answer to q over D w.r.t. $\mathcal{AC}$ is true iff $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$ has no solution.

Computing Consistent Answers - Example

- *BalanceSheets* database with $\mathcal{AC} = \{\kappa_1, \kappa_2, \kappa_3\}$ and
  q_1 : $BalanceSheets(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) \geq 20$
- the cardinality λ of *card*-minimal repairs is 1

$$\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D):$$

$$\begin{cases} \color{red}{1 = \sum_{i=\in\mathcal{I}} \delta_i} \\ z_4 - z_8 = z_9 \\ z_1 + z_9 = z_{10} \\ z_2 + z_3 = z_4 \\ z_5 + z_6 + z_7 = z_8 \\ z_9 \leq 20 \\ w_1 = z_1 - 50 \\ w_2 = z_2 - 100 \end{cases} \quad \begin{array}{l} w_3 = z_3 - 120 \\ w_4 = z_4 - 250 \\ w_5 = z_5 - 120 \\ w_6 = z_6 - 20 \\ w_7 = z_7 - 80 \\ w_8 = z_8 - 220 \\ w_9 = z_9 - 30 \end{array} \quad \begin{array}{l} w_{10} = z_{10} - 80 \\ w_i - M\delta_i \leq 0 \\ -w_i - M\delta_i \leq 0 \\ z_i - M \leq 0 \\ -z_i - M \leq 0 \\ z_i, w_i \in \mathbb{Z} \\ \delta_i \in \{0, 1\} \end{array}$$

Eliminating variables and inequalities

- The size of $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ and $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$ can be reduced in the number of both variables and (in)equalities
- Removing linearly dependent columns of the coefficient matrix $\mathbf{A}|\mathbf{B}$ entails a reduction of variables z_i which in turn implies a reduction of inequalities involving variables δ_i and w_i

$$\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D):$$

$$\begin{cases}
1 = \sum_{i=\in\mathcal{I}} \delta_i & w_3 = z_3 - 120 & w_{10} = z_{10} - 80 \\
z_4 - z_8 = z_9 & w_4 = z_4 - 250 & w_i - M\delta_i \leq 0 \\
z_1 + z_9 = z_{10} & w_5 = z_5 - 120 & -w_i - M\delta_i \leq 0 \\
z_2 + z_3 = z_4 & w_6 = z_6 - 20 & z_i - M \leq 0 \\
z_5 + z_6 + z_7 = z_8 & w_7 = z_7 - 80 & -z_i - M \leq 0 \\
z_9 \leq 20 & w_8 = z_8 - 220 & z_i, w_i \in \mathbb{Z} \\
w_1 = z_1 - 50 & w_9 = z_9 - 30 & \delta_i \in \{0, 1\} \\
w_2 = z_2 - 100 & &
\end{cases}$$

- Linearly dependent columns in $\mathbf{A} \times \vec{z} \leq \mathbf{B}$ must be removed before generating $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ and $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$

Eliminating variables and inequalities

- The size of $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ and $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$ can be reduced in the number of both variables and (in)equalities
- Removing linearly dependent columns of the coefficient matrix **A|B** entails a reduction of variables z_i which in turn implies a reduction of inequalities involving variables δ_i and w_i

$\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$:

$$\begin{cases} 1 = \sum_{i=\in\mathcal{I}} \delta_i & w_3 = z_3 - 120 & w_{10} = z_{10} - 80 \\ z_4 - z_8 = z_9 & w_4 = z_4 - 250 & w_i - M\delta_i \leq 0 \\ z_1 + z_9 = z_{10} & w_5 = z_5 - 120 & -w_i - M\delta_i \leq 0 \\ z_2 + z_3 = z_4 & w_6 = z_6 - 20 & z_i - M \leq 0 \\ z_5 + z_6 + z_7 = z_8 & w_7 = z_7 - 80 & -z_i - M \leq 0 \\ z_9 \leq 20 & w_8 = z_8 - 220 & z_i, w_i \in \mathbb{Z} \\ w_1 = z_1 - 50 & w_9 = z_9 - 30 & \delta_i \in \{0,1\} \\ w_2 = z_2 - 100 & & \end{cases}$$

- Linearly dependent columns in $\mathbf{A} \times \vec{z} \leq \mathbf{B}$ must be removed before generating $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ and $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$

Eliminating variables and inequalities

- The size of $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ and $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$ can be reduced in the number of both variables and (in)equalities
- Removing linearly dependent columns of the coefficient matrix $\mathbf{A}|\mathbf{B}$ entails a reduction of variables z_i which in turn implies a reduction of inequalities involving variables δ_i and w_i

$$\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D):$$

$$\begin{cases}
1 = \sum_{i=\in\mathcal{I}} \delta_i & w_3 = z_3 - 120 & w_{10} = z_{10} - 80 \\
z_4 - z_8 = z_9 & w_4 = z_4 - 250 & w_i - M\delta_i \leq 0 \\
z_1 + z_9 = z_{10} & w_5 = z_5 - 120 & -w_i - M\delta_i \leq 0 \\
z_{2,3} = z_4 & w_6 = z_6 - 20 & z_i - M \leq 0 \\
z_{5,6,7} = z_8 & w_7 = z_7 - 80 & -z_i - M \leq 0 \\
z_9 \leq 20 & w_8 = z_8 - 220 & z_i, w_i \in \mathbb{Z} \\
w_1 = z_1 - 50 & w_9 = z_9 - 30 & \delta_i \in \{0, 1\} \\
w_2 = z_2 - 100 & &
\end{cases}$$

- Linearly dependent columns in $\mathbf{A} \times \vec{z} \leq \mathbf{B}$ must be removed before generating $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ and $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$

Eliminating variables and inequalities

- The size of $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ and $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$ can be reduced in the number of both variables and (in)equalities
- Removing linearly dependent columns of the coefficient matrix $\mathbf{A}|\mathbf{B}$ entails a reduction of variables z_i which in turn implies a reduction of inequalities involving variables δ_i and w_i

$\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$:

$$
\begin{cases}
1 = \sum_{i=\in\mathcal{I}} \delta_i & w_{2,3} = z_2 - 100 - 120 & w_{10} = z_{10} - 80 \\
z_4 - z_8 = z_9 & w_4 = z_4 - 250 & w_i - M\delta_i \leq 0 \\
z_1 + z_9 = z_{10} & w_{5,6,7} = z_5 - 120 - 20 - 80 & -w_i - M\delta_i \leq 0 \\
z_{2,3} = z_4 & w_8 = z_8 - 220 & z_i - M \leq 0 \\
z_{5,6,7} = z_8 & w_9 = z_9 - 30 & -z_i - M \leq 0 \\
z_9 \leq 20 & & z_i, w_i \in \mathbb{Z} \\
w_1 = z_1 - 50 & & \delta_i \in \{0, 1\}
\end{cases}
$$

- Linearly dependent columns in $\mathbf{A} \times \vec{z} \leq \mathbf{B}$ must be removed before generating $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ and $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$

Eliminating variables and inequalities

- The size of $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ and $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$ can be reduced in the number of both variables and (in)equalities
- Removing linearly dependent columns of the coefficient matrix $\mathbf{A}|\mathbf{B}$ entails a reduction of variables z_i which in turn implies a reduction of inequalities involving variables δ_i and w_i

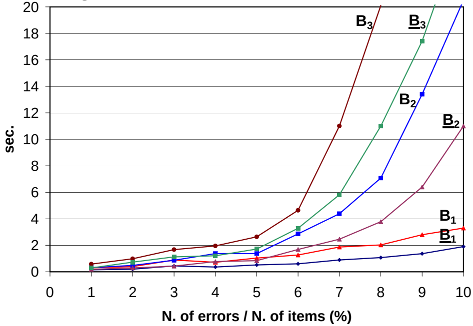
$$\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D):$$

$$\begin{cases} 1 = \sum_{i=\in\mathcal{I}} \delta_i & w_{2,3} = z_2 - 100 - 120 & w_{10} = z_{10} - 80 \\ z_4 - z_8 = z_9 & w_4 = z_4 - 250 & w_i - M\delta_i \leq 0 \\ z_1 + z_9 = z_{10} & w_{5,6,7} = z_5 - 120 - 20 - 80 & -w_i - M\delta_i \leq 0 \\ z_{2,3} = z_4 & w_8 = z_8 - 220 & z_i - M \leq 0 \\ z_{5,6,7} = z_8 & w_9 = z_9 - 30 & -z_i - M \leq 0 \\ z_9 \leq 20 & & z_i, w_i \in \mathbb{Z} \\ w_1 = z_1 - 50 & & \delta_i \in \{0, 1\} \end{cases}$$

- Linearly dependent columns in $\mathbf{A} \times \vec{z} \leq \mathbf{B}$ must be removed before generating $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ and $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$

Experiments on data set *Balance Sheets*

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

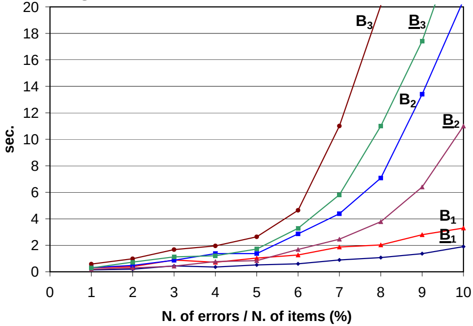


- B_1, B_2, B_3 contains 112, 256, and 378 tuples, respectively
- Lines labeled with B_i and $\underline{B}_i$ refer to basic and reduced-size ILPs, respectively

- the typical number of items in a balance sheet is less than 400 and the typical percentage of errors is less than 5% of acquired data
- In this range, at most 3 seconds are needed to compute CQAs

Experiments on data set *Balance Sheets*

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

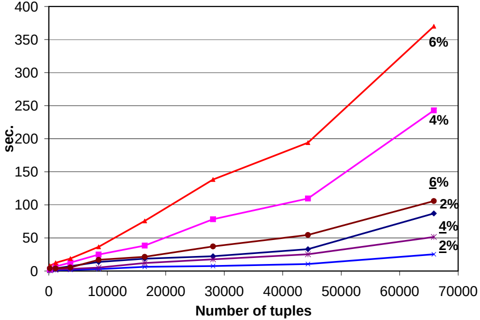


- B_1, B_2, B_3 contains 112, 256, and 378 tuples, respectively
- Lines labeled with B_i and $\underline{B}_i$ refer to basic and reduced-size ILPs, respectively

- the typical number of items in a balance sheet is less than 400 and the typical percentage of errors is less than 5% of acquired data
- In this range, at most 3 seconds are needed to compute CQAs

Experiments on data set *Departmental Projects*

- We analyzed the execution times for CQA on increasing database sizes and different percentages of erroneous values

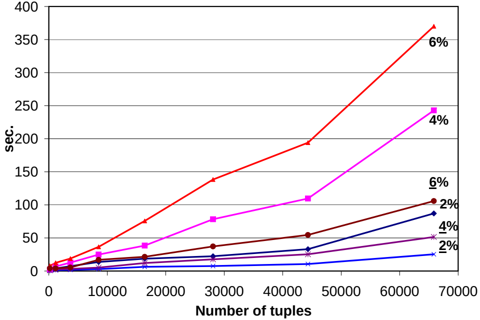


- lines labeled with underlined percentage values refer to reduced-size ILPs

- For the algorithm using reduced-size ILPs the average execution time remains sufficiently small (less than 2 mins and 30 secs for 6% and 2% error rates, respectively)

Experiments on data set *Departmental Projects*

- We analyzed the execution times for CQA on increasing database sizes and different percentages of erroneous values



- lines labeled with underlined percentage values refer to reduced-size ILPs

- For the algorithm using reduced-size ILPs the average execution time remains sufficiently small (less than 2 mins and 30 secs for 6% and 2% error rates, respectively)

Outline

1. Introduction
   - Motivation
   - Previous Work
   - Contribution
2. Preliminaries
   - Aggregate Constraints
   - Repairs
   - Aggregate Queries
3. Query Answering
   - Steady Aggregation Expressions
   - Computing Consistent Answers
   - Reducing the size of ILP
   - Experimental Results
4. Conclusion and Future Work

Conclusion and Future Work

- We have introduced a framework for computing consistent answers to aggregate 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
- Further work will be devoted to devising strategies for computing consistent answers to more expressive forms of queries

Conclusion and Future Work

- We have introduced a framework for computing consistent answers to aggregate 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
- Further work will be devoted to devising strategies for computing consistent answers to more expressive forms of queries

Conclusion and Future Work

- We have introduced a framework for computing consistent answers to aggregate 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
- Further work will be devoted to devising strategies for computing consistent answers to more expressive forms of queries

Conclusion and Future Work

- We have introduced a framework for computing consistent answers to aggregate 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
- Further work will be devoted to devising strategies for computing consistent answers to more expressive forms of queries

Thank you!

... any question?

For Further Reading

Arenas, M., Bertossi, L.E., Chomicki, J.:
Consistent query answers in inconsistent databases.
In: Proc. 18^{th} 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

Papadimitriou, C.H.:
On the complexity of integer programming.
Journal of the Association for Computing Machinery (JACM) Vol. 28(4) (1981) 765–768

Semantics of Aggregate Constraints

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

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

  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

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

  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

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

  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

κ_1 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

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

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

κ_1 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

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

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

κ_1 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

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

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

κ_1 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

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

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

κ_2 for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

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

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

κ_2 for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

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

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

κ_2 for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

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

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**

κ_2 for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

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

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

κ_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

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

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

κ_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

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

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

κ_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

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

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

κ_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

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

Example of Aggregate Queries

- Consider the relation scheme
  BalanceSheets(*Year*, *Section*, *Subsection*, *Type*, *Value*)

q_1 : for each year, is the value of *net cash inflow* greater than 20?

- $\chi_1(x, y) = \langle$*BalanceSheets, Value,* $($*Year*$=x \wedge$*Subsection*$=y$ $)\rangle$
- *BalanceSheets*$(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1,$ 'net cash inflow'$) \geq 20$

Example of Aggregate Queries

- Consider the relation scheme *BalanceSheets*(*Year*, *Section*, *Subsection*, *Type*, *Value*)

q_1 : for each year, is the value of *net cash inflow* greater than 20?

- $\chi_1(x, y) = \langle$*BalanceSheets, Value,* $($*Year*$= x \wedge$*Subsection*$= y)\rangle$
- *BalanceSheets*$(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) \geq 20$

Example of Aggregate Queries

- Consider the relation scheme *BalanceSheets*(*Year*, *Section*, *Subsection*, *Type*, *Value*)

q_1 : for each year, is the value of *net cash inflow* greater than 20?

- $\chi_1(x, y) = \langle$*BalanceSheets, Value,* (*Year*$=x \wedge$*Subsection*$=y$)$\rangle$
- *BalanceSheets*$(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) \geq 20$

Example of Aggregate Queries

- Consider the relation scheme
  BalanceSheets(*Year*, *Section*, *Subsection*, *Type*, *Value*)

q_2 : for years 2008 and 2009, is the sum of *receivables* greater than *payment of accounts*?

- $\chi_3(x) = \langle$*BalanceSheets, Value,* $((Year{=}2008 \vee 2009) \wedge Subsection{=}x)\rangle$
- $BalanceSheets(x_1, x_2, x_3, x_4, x_5) \implies \chi_3(\text{'receivables'}) \geq \chi_3(\text{'payment of accounts'})$

Example of Aggregate Queries

- Consider the relation scheme
  BalanceSheets(*Year*, *Section*, *Subsection*, *Type*, *Value*)

q_2 : for years 2008 and 2009, is the sum of *receivables* greater than *payment of accounts*?

- $\chi_3(x) = \langle \textit{BalanceSheets, Value,} ((\textit{Year}{=}2008 \vee 2009) \wedge \textit{Subsection}{=}x) \rangle$
- $\textit{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_3(\text{'receivables'}) \geq \chi_3(\text{'payment of accounts'})$

Example of Aggregate Queries

- Consider the relation scheme
  BalanceSheets(*Year*, *Section*, *Subsection*, *Type*, *Value*)

q_3 : is the sum of incomings in *cash sales* for both years 2008 and 2009 sufficient to cover the expenses for *long-term financing* of year 2009?

- $\chi_1(x, y) = \langle \textit{BalanceSheets, Value, } (\textit{Year}{=}x \wedge \textit{Subsection}{=}y\,)\rangle$
- $\chi_3(x) = \langle \textit{BalanceSheets, Value, } ((\textit{Year}{=}2008 \vee 2009) \wedge \textit{Subsection}{=}x\,)\rangle$
- $\textit{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_3(\text{'cash sales'}) \geq \chi_1(\text{'long-term financing'}, 2009)$

Example of Aggregate Queries

- Consider the relation scheme
  BalanceSheets(*Year*, *Section*, *Subsection*, *Type*, *Value*)

q_3 : is the sum of incomings in *cash sales* for both years 2008 and 2009 sufficient to cover the expenses for *long-term financing* of year 2009?

- $\chi_1(x, y) = \langle \textit{BalanceSheets, Value, } (\textit{Year}{=}\,x \wedge \textit{Subsection}{=}\,y\,)\rangle$
- $\chi_3(x) = \langle \textit{BalanceSheets, Value, } ((\textit{Year}{=}\,2008 \vee 2009) \wedge \textit{Subsection}{=}\,x\,)\rangle$
- $\textit{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_3(\text{'cash sales'}) \geq \chi_1(\text{'long-term financing'}, 2009)$

Consistent Answers of Aggregate Queries

- We adapt the notion of consistent query answer introduced in [Arenas et Al (PODS 1999)] to our setting
- Let $\rho(D)$ be the database resulting from performing all the updates in the *card*-minimal repair ρ on the database D

Definition (Consistent query answer)

Let $\mathcal{D}$ be a database scheme, D an instance of $\mathcal{D}$, $\mathcal{AC}$ a set of aggregate constraints on $\mathcal{D}$ and q an aggregate query over $\mathcal{D}$. The *consistent query answer* to q on D w.r.t. $\mathcal{AC}$ is true iff, for each *card*-minimal repair ρ for D w.r.t. $\mathcal{AC}$, it holds that $\rho(D) \models q$.

Consistent Answers of Aggregate Queries

- We adapt the notion of consistent query answer introduced in [Arenas et Al (PODS 1999)] to our setting
- Let $\rho(D)$ be the database resulting from performing all the updates in the *card*-minimal repair ρ on the database D

Definition (Consistent query answer)

Let $\mathcal{D}$ be a database scheme, D an instance of $\mathcal{D}$, $\mathcal{AC}$ a set of aggregate constraints on $\mathcal{D}$ and q an aggregate query over $\mathcal{D}$. The *consistent query answer* to q on D w.r.t. $\mathcal{AC}$ is true iff, for each *card*-minimal repair ρ for D w.r.t. $\mathcal{AC}$, it holds that $\rho(D) \models q$.

Two examples of *card*-minimal repairs

Year	*Section*	*Subsection*	*Type*	*Value*	ρ_1	ρ_2
2008	Receipts	beginning cash	drv	50		
2008	Receipts	cash sales	det	100	$\longrightarrow$ 130	
2008	Receipts	receivables	det	120		$\longrightarrow$ 150
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		

- κ_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
- κ_2 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- κ_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*	ρ_1	ρ_2
2008	Receipts	beginning cash	drv	50		
2008	Receipts	cash sales	det	100	$\longrightarrow$ 130	
2008	Receipts	receivables	det	120		$\longrightarrow$ 150
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		

- κ_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
- κ_2 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- κ_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	ρ_1	ρ_2
2008	Receipts	beginning cash	drv	50		
2008	Receipts	cash sales	det	100	$\longrightarrow 130$	
2008	Receipts	receivables	det	120		$\longrightarrow 150$
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		

- κ_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
- κ_2 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- κ_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***	ρ_1	ρ_2
2008	Receipts	beginning cash	drv	50		
2008	Receipts	cash sales	det	100	$\longrightarrow$ 130	
2008	Receipts	receivables	det	120		$\longrightarrow$ 150
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		

- κ_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
- κ_2 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- κ_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***	ρ_1	ρ_2
2008	Receipts	beginning cash	drv	50		
2008	Receipts	cash sales	det	100	$\longrightarrow$ **130**	
2008	Receipts	receivables	det	**120**		$\longrightarrow$ 150
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		

- κ_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
- κ_2 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- κ_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***	ρ_1	ρ_2
2008	Receipts	beginning cash	drv	50		
2008	Receipts	cash sales	det	**100**	$\longrightarrow$ 130	
2008	Receipts	receivables	det	120		$\longrightarrow$ **150**
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		

- κ_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
- κ_2 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- κ_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	ρ_1	ρ_2
2008	Receipts	beginning cash	drv	50		
2008	Receipts	cash sales	det	100	$\longrightarrow$ 130	
2008	Receipts	receivables	det	120		$\longrightarrow$ 150
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		

- κ_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
- κ_2 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*
- κ_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	ρ_1	ρ_2
2008	Receipts	beginning cash	drv	**50**		
2008	Receipts	cash sales	det	100	$\longrightarrow$ 130	
2008	Receipts	receivables	det	120		$\longrightarrow$ 150
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**		

κ_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

κ_2 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

κ_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

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 $R_2(\underline{Project}, Department, Costs\ )$ 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 20*K*)
- 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 α of the aggregation function χ, 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(<u>*Project*</u>, *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 $\chi = \langle R_2, 1, (\text{Costs} \geq 20K) \rangle$
- As attribute *Costs* is a measure attribute of R_2, and it occurs in the formula α of the aggregation function χ, 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 $\chi = \langle R_2, 1, (\text{Costs} \geq 20K)\rangle$
- As attribute *Costs* is a measure attribute of R_2, and it occurs in the formula α of the aggregation function χ, 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 $\chi = \langle R_2, 1, (\text{Costs} \geq 20K)\rangle$
- As attribute *Costs* is a measure attribute of R_2, and it occurs in the formula α of the aggregation function χ, the above-introduced aggregate constraint is not steady (condition (1) of the Definition of steady aggregate constraint is not satisfied).

Constant M (1/2)

- The value of M derives from a well-known general result shown in [Papadimitriou (JACM 1981)] regarding the existence of bounded solutions of systems of linear equalities
- In our case, this result implies that, if the first two (in)equalities of $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$ have at least one solution, then they admit at least one solution where (absolute) values are less than M
- this means that if there is a repair for D w.r.t. $\mathcal{AC}$ then there is an M-bounded repair for D w.r.t. $\mathcal{AC}$ changing the same set of values
- in order to repair card-minimal repairs and consistent answers we can look at M-bounded repairs only

Constant M (1/2)

- The value of M derives from a well-known general result shown in [Papadimitriou (JACM 1981)] regarding the existence of bounded solutions of systems of linear equalities
- In our case, this result implies that, if the first two (in)equalities of $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$ have at least one solution, then they admit at least one solution where (absolute) values are less than M
- this means that if there is a repair for D w.r.t. $\mathcal{AC}$ then there is an M-bounded repair for D w.r.t. $\mathcal{AC}$ changing the same set of values
- in order to repair card-minimal repairs and consistent answers we can look at M-bounded repairs only

Constant *M* (1/2)

- The value of M derives from a well-known general result shown in [Papadimitriou (JACM 1981)] regarding the existence of bounded solutions of systems of linear equalities
- In our case, this result implies that, if the first two (in)equalities of $\mathcal{ILP}(\mathcal{D}, \mathcal{AC}, D)$ have at least one solution, then they admit at least one solution where (absolute) values are less than M
- this means that if there is a repair for D w.r.t. $\mathcal{AC}$ then there is an M-bounded repair for D w.r.t. $\mathcal{AC}$ changing the same set of values
- in order to repair card-minimal repairs and consistent answers we can look at M-bounded repairs only

Constant M (2/2)

- Given a database scheme $\mathcal{D}$, a set $\mathcal{E}$ of steady aggregation expressions on $\mathcal{D}$, and an instance D of $\mathcal{D}$, $\mathcal{ILP}(\mathcal{D}, \mathcal{E}, D)$ is an ILP of the form:

$$\begin{cases} \mathbf{A} \times \vec{z} \leq \mathbf{B}; & & \\ w_i = z_i - v_i & \forall\, i \in \mathcal{I}; & \\ z_i - M \leq 0; & -z_i - M \leq 0; & \forall\, i \in \mathcal{I} \\ w_i - M\delta_i \leq 0; & -w_i - M\delta_i \leq 0; & \forall\, i \in \mathcal{I}; \\ z_i, w_i \in \mathbb{Z}; & \delta_i \in \{0, 1\}; & \forall\, i \in \mathcal{I}; \end{cases}$$

- $M = n \cdot (ma)^{2m+1}$, where: a is the maximum among the modules of the coefficients in $\mathbf{A}$ and of the values v_i, and $m = |\mathcal{I}| + r$, and $n = 2 \cdot |\mathcal{I}| + r$, where r is the number of rows of $\mathbf{A}$
- The size of M is polynomial in the size of the database, as it is bounded by $\log n + (2 \cdot m + 1) \cdot \log(ma)$

Constant M (2/2)

- Given a database scheme $\mathcal{D}$, a set $\mathcal{E}$ of steady aggregation expressions on $\mathcal{D}$, and an instance D of $\mathcal{D}$, $\mathcal{ILP}(\mathcal{D}, \mathcal{E}, D)$ is an ILP of the form:

$$\begin{cases} \mathbf{A} \times \vec{z} \leq \mathbf{B}; & & \\ w_i = z_i - v_i & \forall\, i \in \mathcal{I}; & \\ z_i - M \leq 0; & -z_i - M \leq 0; & \forall\, i \in \mathcal{I} \\ w_i - M\delta_i \leq 0; & -w_i - M\delta_i \leq 0; & \forall\, i \in \mathcal{I}; \\ z_i, w_i \in \mathbb{Z}; & \delta_i \in \{0, 1\}; & \forall\, i \in \mathcal{I}; \end{cases}$$

- $M = n \cdot (ma)^{2m+1}$, where: a is the maximum among the modules of the coefficients in $\mathbf{A}$ and of the values v_i, and $m = |\mathcal{I}| + r$, and $n = 2 \cdot |\mathcal{I}| + r$, where r is the number of rows of $\mathbf{A}$
- The size of M is polynomial in the size of the database, as it is bounded by $\log n + (2 \cdot m + 1) \cdot \log(ma)$

Eliminating variables and inequalities (2)

- Both these ILP problems consist of $\mathbf{A} \times \vec{z} \leq \mathbf{B}$ augmented with further inequalities involving new variables δ_i and w_i
- The number of these variables and inequalities depends on the number of variables z_i occurring in $\mathbf{A} \times \vec{z} \leq \mathbf{B}$
- The elimination of linearly dependent columns yields no reduction of size when applied on the whole coefficient matrixes of $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ or $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$
- The inequalities different from $\mathbf{A} \times \vec{z} \leq \mathbf{B}$ make all the columns of the coefficient matrixes linearly independent
- It is mandatory that linearly dependent columns in $\mathbf{A} \times \vec{z} \leq \mathbf{B}$ are removed before generating $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ and $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$.

Eliminating variables and inequalities (2)

- Both these ILP problems consist of $\mathbf{A} \times \vec{z} \leq \mathbf{B}$ augmented with further inequalities involving new variables δ_i and w_i
- The number of these variables and inequalities depends on the number of variables z_i occurring in $\mathbf{A} \times \vec{z} \leq \mathbf{B}$
- The elimination of linearly dependent columns yields no reduction of size when applied on the whole coefficient matrixes of $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ or $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$
- The inequalities different from $\mathbf{A} \times \vec{z} \leq \mathbf{B}$ make all the columns of the coefficient matrixes linearly independent
- It is mandatory that linearly dependent columns in $\mathbf{A} \times \vec{z} \leq \mathbf{B}$ are removed before generating $\mathcal{OPT}(\mathcal{D}, \mathcal{AC}, D)$ and $\mathcal{CQAP}(\mathcal{D}, \mathcal{AC}, q, D)$.

Experiment Setting

- We experimentally validated our framework for computing consistent answers on two data sets
  - *Balance Sheets*, containing real-life balance-sheet data
  - *Departmental Projects*, synthetic data set containing information about projects developed in different departments
- We used LINDO API 4.0 as ILP solver, and a PC with Intel Pentium 4 Processor at 3.00 GHz and 4GB RAM

Experiment Setting

- We experimentally validated our framework for computing consistent answers on two data sets
  - *Balance Sheets*, containing real-life balance-sheet data
  - *Departmental Projects*, synthetic data set containing information about projects developed in different departments
- 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 $\mathcal{AC} = \{\kappa_1, \kappa_2, \kappa_3\}$ and the queries q_1, q_2, q_3

κ_1 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

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

κ_2 for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

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

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

- We considered the aggregate constraints $\mathcal{AC} = \{\kappa_1, \kappa_2, \kappa_3\}$ and the queries q_1, q_2, q_3

κ_1 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

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

κ_2 for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

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

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

- We considered the aggregate constraints $\mathcal{AC} = \{\kappa_1, \kappa_2, \kappa_3\}$ and the queries q_1, q_2, q_3

κ_1 for each year, the *net cash inflow* must be equal to the difference between *total cash receipts* and *total disbursements*

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

κ_2 for each year, the *ending cash balance* must be equal to the sum of the *beginning cash* and the *net cash inflow*.

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

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

κ_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

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

q_1 : for each year, is the value of *net cash inflow* greater than 20?

- $\chi_1(x, y) = \langle BalanceSheets, Value, (Year{=}x \wedge Subsection{=}y) \rangle$
- $BalanceSheets(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) \geq 20$

q_2 : for years 2008 and 2009, is the sum of *receivables* greater than *payment of accounts*?

- $\chi_3(x) = \langle BalanceSheets, Value, ((Year{=}2008 \vee 2009) \wedge Subsection{=}x) \rangle$
- $BalanceSheets(x_1, x_2, x_3, x_4, x_5) \implies \chi_3(\text{'receivables'}) \geq \chi_3(\text{'payment of accounts'})$

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

κ_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

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

q_1 : for each year, is the value of *net cash inflow* greater than 20?

- $\chi_1(x, y) = \langle \mathit{BalanceSheets}, \mathit{Value}, (\mathit{Year}{=}x \wedge \mathit{Subsection}{=}y) \rangle$
- $\mathit{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) \geq 20$

q_2 : for years 2008 and 2009, is the sum of *receivables* greater than *payment of accounts*?

- $\chi_3(x) = \langle \mathit{BalanceSheets}, \mathit{Value}, ((\mathit{Year}{=}2008 \vee 2009) \wedge \mathit{Subsection}{=}x) \rangle$
- $\mathit{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_3(\text{'receivables'}) \geq \chi_3(\text{'payment of accounts'})$

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

κ_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

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

q_1 : for each year, is the value of *net cash inflow* greater than 20?

- $\chi_1(x, y) = \langle \mathit{BalanceSheets}, \mathit{Value}, (\mathit{Year}{=}x \wedge \mathit{Subsection}{=}y) \rangle$
- $\mathit{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_1(x_1, \text{'net cash inflow'}) \geq 20$

q_2 : for years 2008 and 2009, is the sum of *receivables* greater than *payment of accounts*?

- $\chi_3(x) = \langle \mathit{BalanceSheets}, \mathit{Value}, ((\mathit{Year}{=}2008 \vee 2009) \wedge \mathit{Subsection}{=}x) \rangle$
- $\mathit{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_3(\text{'receivables'}) \geq \chi_3(\text{'payment of accounts'})$

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

q_3 : is the sum of incomings in *cash sales* for both years 2008 and 2009 sufficient to cover the expenses for *long-term financing* of year 2009?

- $\chi_1(x, y) = \langle \mathit{BalanceSheets, Value,} \ (\mathit{Year} = x \wedge \mathit{Subsection} = y) \rangle$
- $\chi_3(x) = \langle \mathit{BalanceSheets, Value,} \ ((\mathit{Year} = 2008 \vee 2009) \wedge \mathit{Subsection} = x) \rangle$
- $\mathit{BalanceSheets}(x_1, x_2, x_3, x_4, x_5) \implies \chi_3(\text{'cash sales'}) \geq \chi_1(\text{'long-term financing'}, 2009)$

Constraints and Queries of Experiments on data set Departmental Projects (1/4)

- We considered the following database scheme $\mathcal{D}$:

- *Project(Name, Department, Funding)*
- *Expense(Project, Description, Type, Date, Amount)*
- *Department(Name, TotalFunding)*
- *MaxExpense(Type, Department, Threshold)*

where underlined attributes denote keys, and measure attributes are as follows: $\mathcal{M}_{Project} = \{Funding\}$, $\mathcal{M}_{Department} = \{TotalFunding\}$, $\mathcal{M}_{MaxExpense} = \{Threshold\}$, $\mathcal{M}_{Expense} = \{Amount\}$.

Constraints and Queries of Experiments on data set Departmental Projects (2/4)

We considered the following set of aggregate constraints $\mathcal{AC}$:

1) $Project(x, \_\_) \implies \chi_1(x) - \chi_2(x) \geq 0$,
   where
   - $\chi_1(x) = \langle\ Project, Funding, (Name{=}x)\rangle$
   - $\chi_2(x) = \langle\ Expense, Amount, (Project{=}x)\rangle$.

   This constraint imposes that the funding for each project must be greater than or equal to the total expenses for the same project.

Constraints and Queries of Experiments on data set Departmental Projects (3/4)

We considered the following set of aggregate constraints $\mathcal{AC}$:

2) $Department(x, \_\_) \implies \chi_3(x) - \chi_4(x) = 0$
where
- $\chi_3(x) = \langle$ *Department*, *TotalFunding*, $(Name = x)\rangle$ and
- $\chi_4(x) = \langle$ *Project*, *Funding*, $(Department = x)\rangle$

This constraint imposes that for each department, the total amount of funding allocated for developing all its projects must be equal to the sum of funding allocated for every single project in the same department

Constraints and Queries of Experiments on data set Departmental Projects (4/4)

3) $Project(x, y, \_)$,
$MaxExpense(z, y, \_) \implies \chi_5(x, z) - \chi_6(z, y) \leq 0$,
where

- $\chi_5(x, z) = \langle$ *Expense*, *Amount*, $(Project{=}x \wedge Type{=}\ z)\rangle$, and
  $\chi_6(z, y) = \langle$ *MaxExpense*, *Threshold*, $(Type{=}z \wedge Department{=}\ y)\rangle$

This constraint imposes that, for each project x developed in a department y, and for each type of expense z which is bounded for department y by the threshold τ, the total amount of expenses of type z for project x must not be greater than τ.

Complexity Classes

- *PTIME*: 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;
- Δ_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 II

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