# DART:
# a Data Acquisition and Repairing Tool

Bettina Fazzinga, Sergio Flesca, Filippo Furfaro and Francesco Parisi

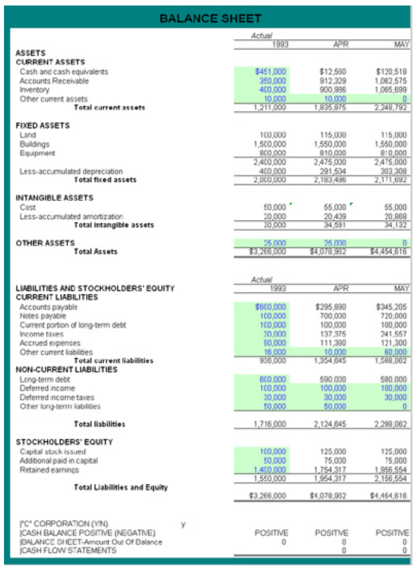
D.E.I.S.
Università della Calabria

{bfazzinga, flesca, furfaro, fparisi}@deis.unical.it

International Workshop on Inconsistency and Incompleteness in Databases
*March 26, 2006 - Munich (Germany)*

# Motivation

- Error-free acquisition of data is mandatory in several application scenarios
  - balance sheet analysis



**BALANCE SHEET**

| | Actual 1993 | APR | MAY |
|---|---|---|---|
| **ASSETS** | | | |
| **CURRENT ASSETS** | | | |
| Cash and cash equivalents | $451,000 | $12,580 | $120,518 |
| Accounts Receivable | 350,000 | 912,329 | 1,062,575 |
| Inventory | 400,000 | 900,986 | 1,065,699 |
| Other current assets | 10,000 | 10,000 | 0 |
| **Total current assets** | 1,211,000 | 1,835,875 | 2,248,792 |
| **FIXED ASSETS** | | | |
| Land | 100,000 | 115,000 | 115,000 |
| Buildings | 1,500,000 | 1,550,000 | 1,550,000 |
| Equipment | 800,000 | 810,000 | 810,000 |
| | 2,400,000 | 2,475,000 | 2,475,000 |
| Less-accumulated depreciation | 400,000 | 291,504 | 303,308 |
| **Total fixed assets** | 2,000,000 | 2,183,496 | 2,171,692 |
| **INTANGIBLE ASSETS** | | | |
| Cost | 50,000 | 55,000 | 55,000 |
| Less-accumulated amortization | 20,000 | 20,409 | 20,868 |
| **Total intangible assets** | 30,000 | 34,591 | 34,132 |
| **OTHER ASSETS** | 25,000 | 25,000 | 0 |
| **Total Assets** | $3,266,000 | $4,078,982 | $4,454,616 |

| | Actual 1993 | APR | MAY |
|---|---|---|---|
| **LIABILITIES AND STOCKHOLDERS' EQUITY** | | | |
| **CURRENT LIABILITIES** | | | |
| Accounts payable | $600,000 | $295,890 | $345,205 |
| Notes payable | 100,000 | 700,000 | 720,000 |
| Current portion of long-term debt | 100,000 | 100,000 | 100,000 |
| Income taxes | 30,000 | 137,375 | 241,557 |
| Accrued expenses | 90,000 | 111,380 | 121,300 |
| Other current liabilities | 16,000 | 10,000 | 60,000 |
| **Total current liabilities** | 936,000 | 1,354,645 | 1,588,062 |
| **NON-CURRENT LIABILITIES** | | | |
| Long-term debt | 600,000 | 590,000 | 580,000 |
| Deferred income | 100,000 | 100,000 | 100,000 |
| Deferred income taxes | 30,000 | 30,000 | 30,000 |
| Other long-term liabilities | 50,000 | 50,000 | 0 |
| **Total liabilities** | 1,716,000 | 2,124,645 | 2,298,062 |
| **STOCKHOLDERS' EQUITY** | | | |
| Capital stock issued | 100,000 | 125,000 | 125,000 |
| Additional paid in capital | 50,000 | 75,000 | 75,000 |
| Retained earnings | 1,400,000 | 1,754,317 | 1,956,554 |
| | 1,550,000 | 1,954,317 | 2,156,554 |
| **Total Liabilities and Equity** | $3,266,000 | $4,078,982 | $4,454,616 |
| ["C" CORPORATION (Y/N) y | | | |
| [CASH BALANCE POSITIVE (NEGATIVE) | POSITIVE | POSITIVE | POSITIVE |
| [BALANCE SHEET-Amount Out Of Balance | 0 | 0 | 0 |
| [CASH FLOW STATEMENTS | | 0 | 0 |

**electronic doc**

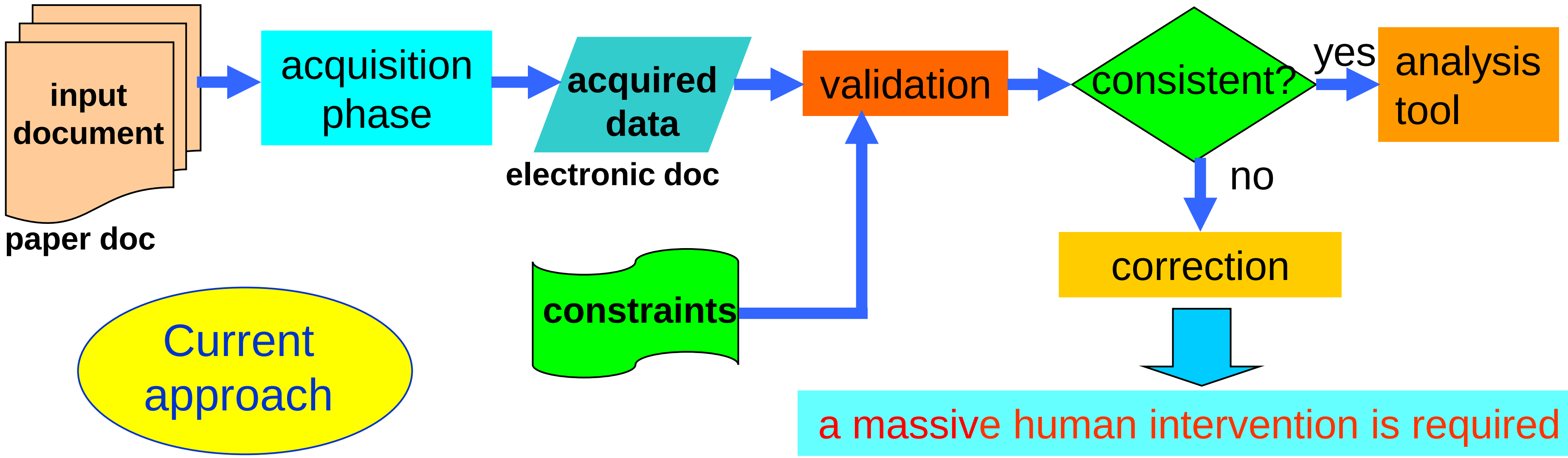
Balance sheet analysis tool


**analysis report**

- generally balance sheets are available as paper documents, thus they cannot be processed by balance analysis tools, since these work only on electronic data

# Motivation

- Error-free acquisition of data is mandatory in several application scenarios
  - balance sheet analysis




- currently, integrity constraints defined on the input data are exploited only for validating acquired data
- if data are inconsistent all the document portions involved into unsatisfied constraint must be checked for locating and correcting errors

# Motivation

- For instance

OCR tool

**source document**

| | |
|---|---|
| cash sales | *100* |
| receivables | *120* |
| total cash receipts | *220* |
| payment of accounts | *120* |
| long-term financing | *40* |
| total disbursements | *160* |
| net cash inflow | *60* |

**100 +**
**120 =**
**220**

**120 +**
**40 =**
**160**

**220 -**
**160 =**
**60**

**acquired document**

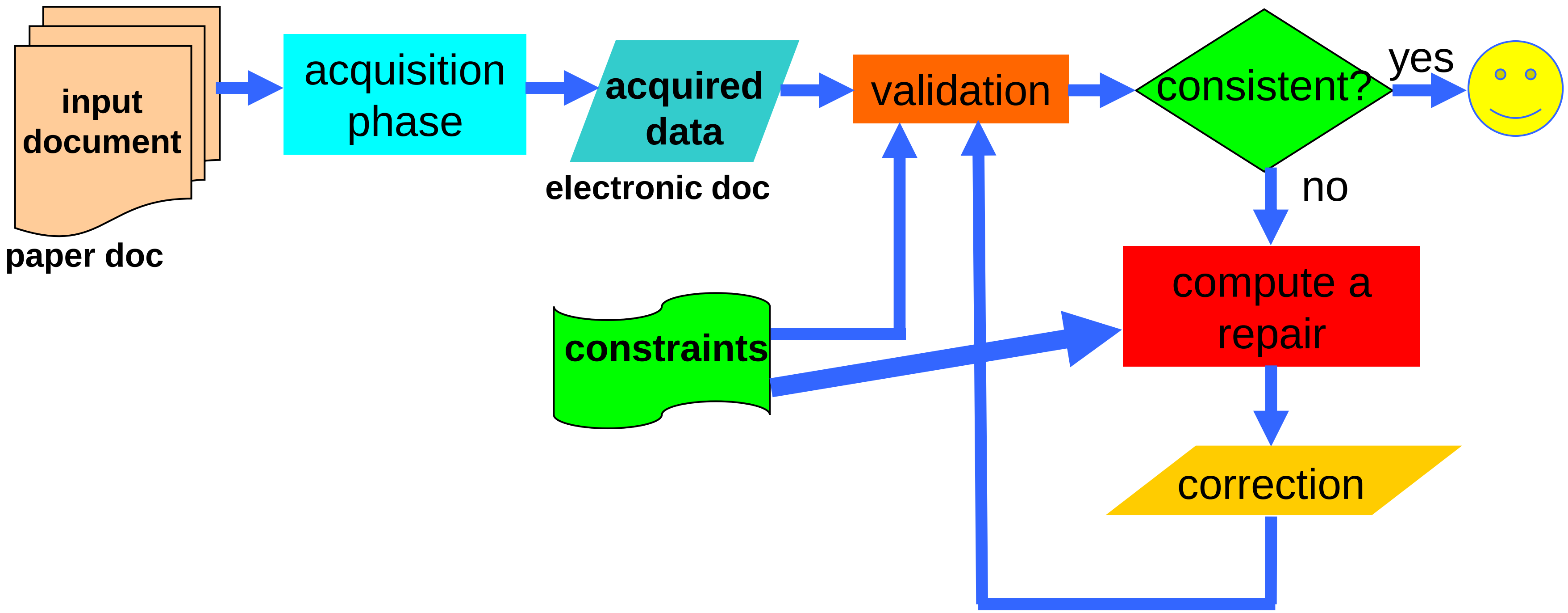
| | | |
|---|---|---|
| cash sales | *100* | ✓ |
| receivables | *120* | ✓ |
| total cash receipts | *250* | ✓ |
| payment of accounts | *120* | ✓ |
| long-term financing | *40* | ✓ |
| total disbursement | *160* | ✓ |
| net cash inflow | *60* | ✓ |

a massive human intervention is required for correcting errors

- constraints like those defined in the context of balance-sheet data can be express by aggregate constraints

# Key Idea

exploit integrity constraints for **suggesting corrections**




the human intervention will be limited to verify only located suggestions

# Key Idea

**exploit integrity constraints for suggesting corrections**

- For instance

**acquired document**

| | |
|---|---|
| cash sales | *100* |
| receivables | *120* |
| total cash receipts | *250* |
| payment of accounts | *120* |
| long-term financing | *40* |
| total disbursement | *160* |
| net cash inflow | *60* |

DART suggests decreasing the value down to 220

- in this case the operator will have to verify a single value instead of all the values in the table

# Outline

- Repairing strategies
- DART architecture
- Aggregate constraints
- Steady aggregate constraints (SAC)
- Computing a card-minimal repair

# Repairing strategy

- What is a reasonable strategy for repairing the acquired data?

~~**Tuple deletion / insertion**~~

**The inconsistent cash budget**

| | | |
|---|---|---|
| **Receipts** | cash sales | 100 |
| | receivables | 120 |
| | ***total cash*** | ***250*** |

**100 + 120 ≠ 250**

**The repaired cash budget**

| | | |
|---|---|---|
| **Receipts** | cash sales | 100 |
| | receivables | 120 |
| | **XXXXX** | **30** |
| | ***total cash*** | ***250*** |

**100 +**
**120 +**
**30 =**
**250**

Adding a new tuple means that the OCR tool skipped a whole row when acquiring ... ***It's rather unrealistic!!!***

# Repairing strategy

- What is a reasonable strategy for repairing the acquired data?
- The most natural approach is updating directly the numerical data
  - Work at attribute-level, rather than tuple-level

**The inconsistent cash budget**

| | | |
|---|---|---|
| **Receipts** | cash sales | 100 |
| | receivables | 120 |
| | ***total cash*** | ***250*** |

**100 + 120 ≠ 250**

**The repaired cash budget**

| | | |
|---|---|---|
| **Receipts** | cash sales | 100 |
| | receivables | 120 |
| | ***total cash*** | ***220*** |

**100 +**
**120 =**
**220**

- In our context, we can reasonably assume that inconsistencies are due to symbol recognition errors
- Thus, trying to re-construct the actual data values (without changing the number of tuples) is well founded

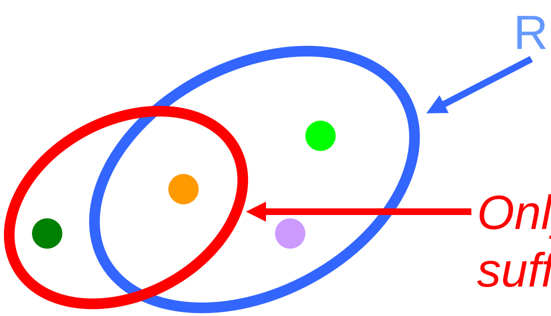
# *Card-minimal* semantics

The most probable case is that the acquiring system made the minimum number of errors

***Card-minimal semantics***

It means assuming that the minimum number of errors occurred

A repair R is *card*-minimal for D iff there is no repair R' for D consisting of fewer updates than R

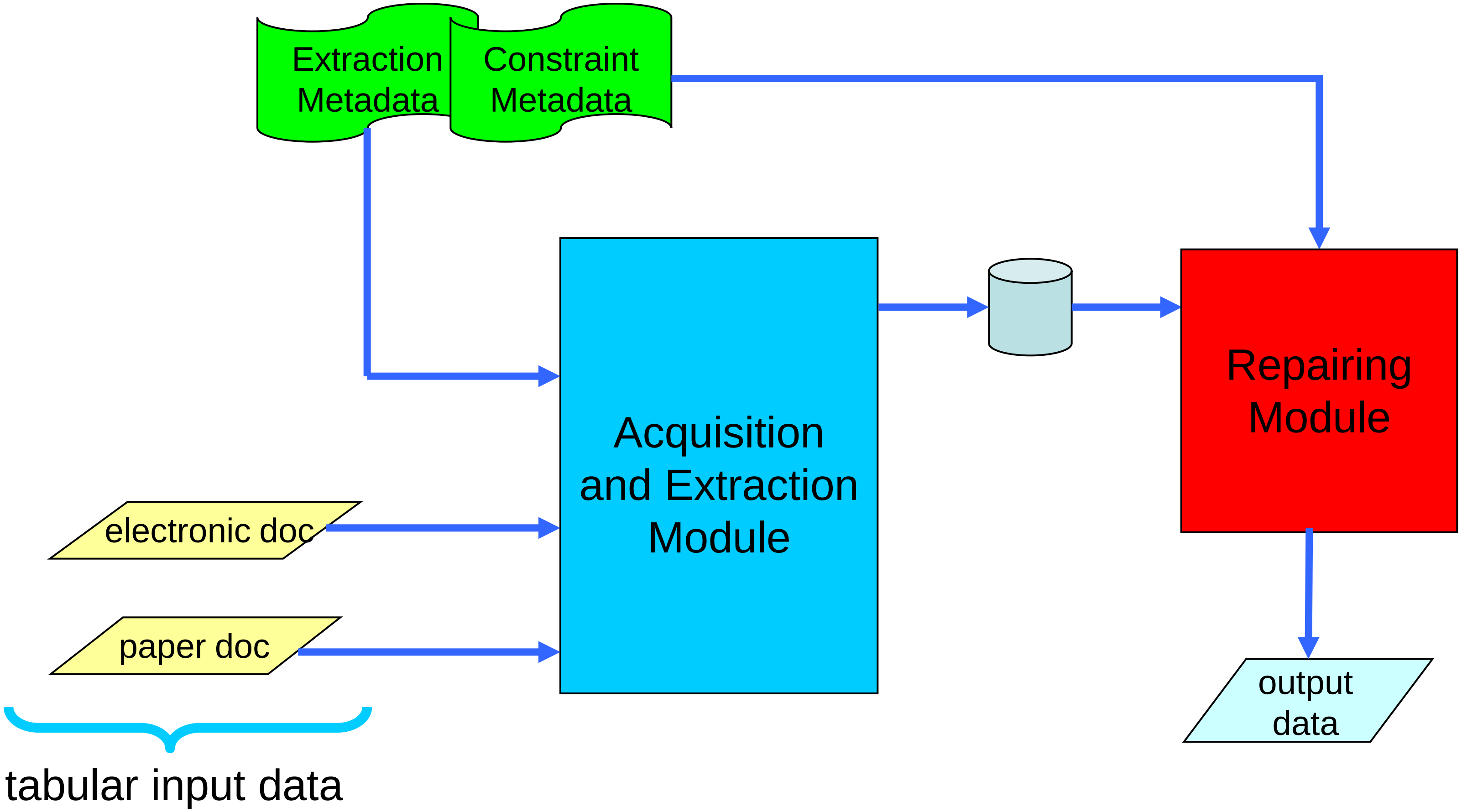



*Only two updates do not suffice to repair D!*

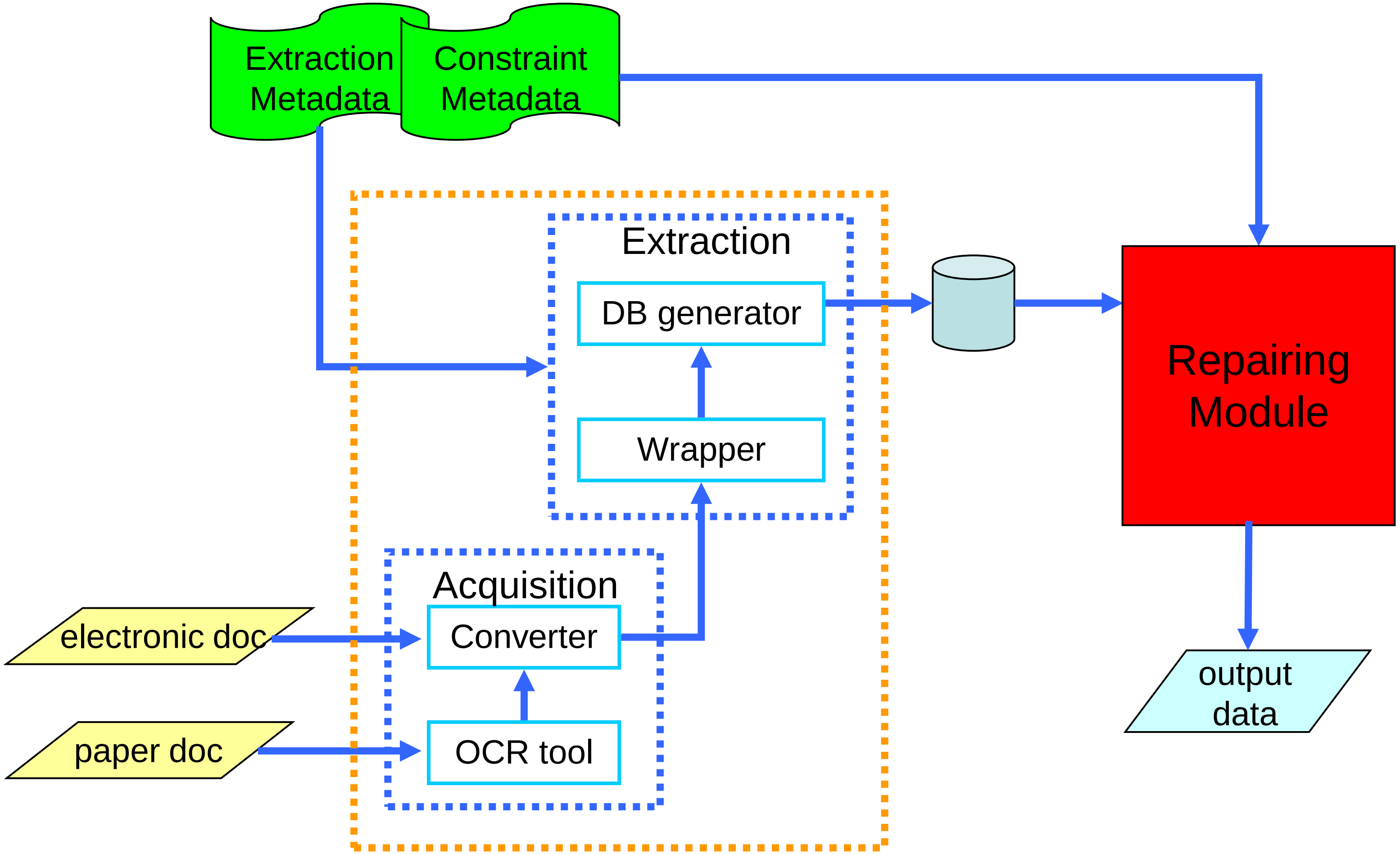
# Outline

- Repairing strategies
- DART architecture
- Aggregate constraints
- Steady aggregate constraints (SAC)
- Computing a card-minimal repair

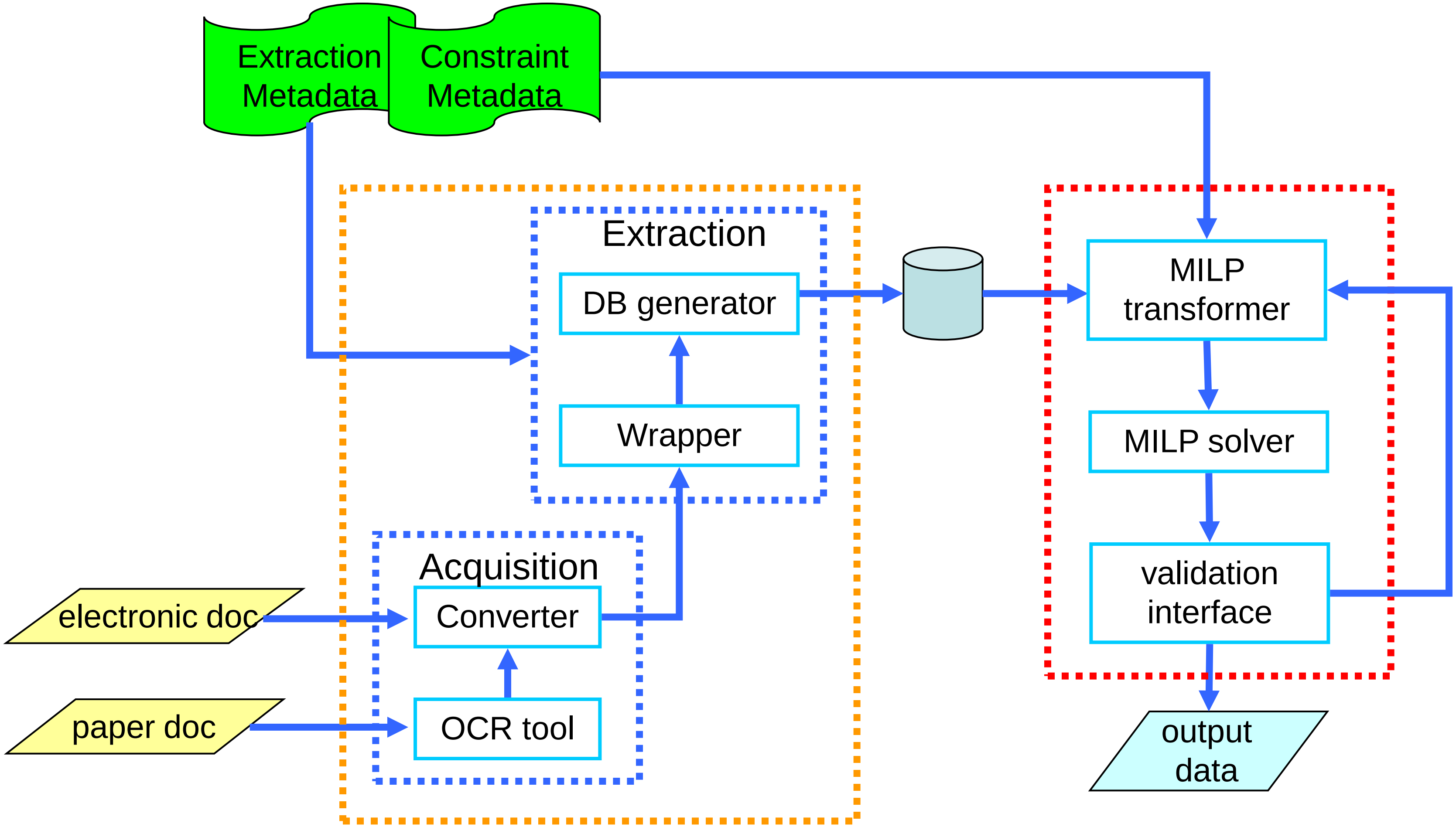
# DART architecture



# DART architecture - Acquisition and Extraction Module



# DART architecture - Repairing Module



# Outline

- Repairing strategies
- DART architecture
- Aggregate constraints
- Steady aggregate constraints (SAC)
- Computing a card-minimal repair

# Aggregate constraints: the application context

- A cash budget for a firm:

Sections

Subsections

| Year | | 2004 |
|---|---|---|
| **Receipts** | | |
| | beginning cash | *20* |
| | cash sales | *100* |
| | receivables | *120* |
| | total cash receipts | *220* |
| **Disbursements** | | |
| | payment of accounts | *120* |
| | capital expenditure | *0* |
| | long-term financing | *40* |
| | total disbursements | *160* |
| **Balance** | | |
| | net cash inflow | *60* |
| | ending cash balance | *80* |

aggregate items are obtained by aggregating detail items of the same section

# Aggregate constraints: the application context

- A cash budget for a firm:

Sections

Subsections

| Year | | 2004 |
|---|---|---|
| **Receipts** | | |
| | beginning cash | *20* |
| | cash sales | *100* |
| | receivables | *120* |
| | total cash receipts | *220* |
| **Disbursements** | | |
| | payment of accounts | *120* |
| | capital expenditure | *0* |
| | long-term financing | *40* |
| | total disbursements | *160* |
| **Balance** | | |
| | net cash inflow | *60* |
| | ending cash balance | *80* |

derived items are obtained using the value of other item of any type and belonging to any section

# Aggregate constraints: the application context

- A cash budget satisfy some integrity constraints:

| Year | | 2004 |
|---|---|---|
| **Receipts** | | |
| | beginning cash | *20* |
| | cash sales | *100* |
| | receivables | *120* |
| | total cash receipts | *220* |
| **Disbursements** | | |
| | payment of accounts | *120* |
| | capital expenditure | *0* |
| | long-term financing | *40* |
| | total disbursements | *160* |
| **Balance** | | |
| | net cash inflow | *60* |
| | ending cash balance | *80* |

$$100 + 120 = 220$$

$$120 + 0 + 40 = 160$$

1) for each section, the sum of all detail items must be equal to the value of the aggregate item

# Aggregate constraints: the application context

- A cash budget satisfy some integrity constraints:

| Year | | 2004 |
|---|---|---|
| **Receipts** | | |
| | beginning cash | *20* |
| | cash sales | *100* |
| | receivables | *120* |
| | total cash receipts | *220* |
| **Disbursements** | | |
| | payment of accounts | *120* |
| | capital expenditure | *0* |
| | long-term financing | *40* |
| | total disbursements | *160* |
| **Balance** | | |
| | net cash inflow | *60* |
| | ending cash balance | *80* |

**220 -**
**160 =**
**60**

2)

the net cash inflow must be equal to the difference between total cash receipts and total disbursements

# From the paper document to its digitized version

| Year | | 2004 |
|---|---|---|
| **Receipts** | | |
| | beginning cash | *20* |
| | cash sales | *100* |
| | receivables | *120* |
| | total cash receipts | *220* |
| **Disbursements** | | |
| | payment of accounts | *120* |
| | capital expenditure | *0* |
| | long-term financing | *40* |
| | total disbursements | *160* |
| **Balance** | | |
| | net cash inflow | *60* |
| | ending cash balance | *80* |

CashBudget

| Section | Subsection | Type | Value |
|---|---|---|---|
| Receipts | beginning cash | *drv* | *20* |
| Receipts | cash sales | *det* | *100* |
| Receipts | receivables | *det* | *120* |
| Receipts | total cash receipts | *aggr* | *250* |
| Disbursements | payment of accounts | *det* | *120* |
| Disbursements | capital expenditure | *det* | *0* |
| Disbursements | long-term financing | *det* | *40* |
| Disbursements | total disbursements | *aggr* | *160* |
| Balance | net cash inflow | *drv* | *60* |
| Balance | ending cash balance | *drv* | *80* |

# Aggregate constraints

- can express constraints like those defined in the context of balance-sheet data

$$\forall \overline{X} \ \left(\phi(\overline{X}) \implies AggrF(\overline{X}) \leq K\right)$$

where:

1. $\phi(\overline{X})$ is a conjunction of atoms
2. $K$ is a constant
3. The aggregation formula $AggrF(\overline{X})$ is the linear combination of aggregation functions

$$\sum_{i=1}^{n} c_i \cdot \chi_i(\overline{X_i})$$

with $\overline{X_i} \subseteq \overline{X}$

# Aggregation function

- Relational scheme $R(A_1, A_2, \ldots A_n)$
  - Measure attributes: numerical attributes representing measures
    - Such as weight, length, price, etc.

Linear combination of attributes

- Aggregation function

$$\chi(x_1, \ldots, x_k) = SELECT\; sum(e) \\ FROM \quad R \\ WHERE\; \alpha(x_1, \ldots, x_k)$$

Boolean formula on constants and attributes of R

# Aggregate constraints

- CashBudget(Section,Subsection,Type,Value)

| Section | Subsection | Type | Value |
|---|---|---|---|
| Receipts | beginning cash | *drv* | *20* |
| Receipts | cash sales | *det* | *100* |
| Receipts | receivables | *det* | *120* |
| Receipts | total cash receipts | *aggr* | *250* |
| Disbursements | payment of accounts | *det* | *120* |
| Disbursements | capital expenditure | *det* | *0* |
| Disbursements | long-term financing | *det* | *40* |
| Disbursements | total disbursements | *aggr* | *160* |
| Balance | net cash inflow | *drv* | *60* |
| Balance | ending cash balance | *drv* | *80* |

1)

for each section, the sum of all detail items must be equal to the value of the aggregate item

Aggregation function:

$$\chi_1(s,t) = SELECT\ sum(Value)\ FROM\ CashBudget\ WHERE\ Section = s\ AND\ Type = t$$

Aggregate constraint:

$$CashBudget(s,\_,\_,\_) \implies \chi_1(s,det) - \chi_1(s,aggr) = 0$$

# Aggregate constraints

- CashBudget(Section,Subsection,Type,Value)

| Section | Subsection | Type | Value |
|---|---|---|---|
| Receipts | beginning cash | *drv* | *20* |
| Receipts | cash sales | *det* | *100* |
| Receipts | receivables | *det* | *120* |
| Receipts | total cash receipts | *aggr* | *250* |
| Disbursements | payment of accounts | *det* | *120* |
| Disbursements | capital expenditure | *det* | *0* |
| Disbursements | long-term financing | *det* | *40* |
| Disbursements | total disbursements | *aggr* | *160* |
| Balance | net cash inflow | *drv* | *60* |
| Balance | ending cash balance | *drv* | *80* |

2)

the net cash inflow must be equal to the difference between total cash receipts and total disbursements

Aggregation function:

$$\chi_2(ss) = SELECT\ sum(Value)\ FROM\ CashBudget\ WHERE\ Subsection = ss$$

Aggregation constraint:

$$CashBudget(\_,\_,\_,\_) \implies \chi_2(net\ cash\ inflow) - [\chi_2(total\ cash\ receipts) - \chi_2(total\ disbursements)] = 0$$

# Outline

- Repairing strategies
- DART architecture
- Aggregate constraints
- Steady aggregate constraints (SACs)
- Computing a card-minimal repair

# Steady aggregate constraints (SACs)

- a restricted form of aggregate constraints
- computing a card-minimal repair w.r.t. a set of SAC can be accomplished by solving an instance of MILP problem

CashBudget

| Section | Subsection | Type | Value | |
|---|---|---|---|---|
| Receipts | beginning cash | *drv* | *20* | |
| Receipts | cash sales | *det* | *100* | → $z_1$ |
| Receipts | receivables | *det* | *120* | → $z_2$ |
| Receipts | total cash receipts | *aggr* | *250* | → $z_3$ |
| Disbursements | payment of accounts | *det* | *120* | $z_4$ |
| Disbursements | capital expenditure | *det* | *0* | $z_5$ |
| Disbursements | long-term financing | *det* | *40* | $z_6$ |
| Disbursements | total disbursements | *aggr* | *160* | → $z_7$ |
| Balance | net cash inflow | *drv* | *60* | |
| Balance | ending cash balance | *drv* | *80* | |

$$\begin{cases} z_1 + z_2 = z_3 \\ z_4 + z_5 + z_6 = z_7 \end{cases}$$

a system of inequalities can be associated if values "involved" in the constraints are independent on repairs

$$CashBudget(s,\_,\_,\_) \implies \chi_1(s, det) - \chi_1(s, aggr) = 0$$

# Steady aggregate constraints (SACs)

An aggregate constraint is an SAC if:

1) no attributes in the WHERE clause are measure attributes
2) no attributes corresponding to variables in the WHERE clause are measure attributes
3) no attributes corresponding to variables shared by two atoms are measure attributes

- CashBudget(Section,Subsection,Type,Value)

$$CashBudget(s, ss, t, v) \implies \chi_1(s, det) - \chi_1(s, aggr) = 0$$

where: $$\begin{aligned} \chi_1(s,t) = & SELECT \ sum(Value) \\ & FROM \ CashBudget \\ & WHERE \ Section = s \\ & AND \ Type = t \end{aligned}$$

# Steady aggregate constraints (SACs)

An aggregate constraint is an SAC if:

1) no attributes in the WHERE clause are measure attributes
2) no attributes corresponding to variables in the WHERE clause are measure attributes
3) no attributes corresponding to variables shared by two atoms are measure attributes

- CashBudget(Section,Subsection,Type,Value)

$$CashBudget(s, ss, t, v) \implies \chi_1(s, det) - \chi_1(s, aggr) = 0$$

where: $$\begin{aligned} \chi_1(s,t) = & SELECT \ sum(Value) \\ & FROM \ \ CashBudget \\ & WHERE \ Section = s \\ & \quad AND \ Type = t \end{aligned}$$

# Steady aggregate constraints (SACs)

An aggregate constraint is an SAC if:

1) no attributes in the WHERE clause are measure attributes
2) no attributes corresponding to variables in the WHERE clause are measure attributes
3) no attributes corresponding to variables shared by two atoms are measure attributes

- CashBudget(Section,Subsection,Type,Value)

$$CashBudget(s, ss, t, v) \implies \chi_1(s, det) - \chi_1(s, aggr) = 0$$

where:
$$\begin{array}{rl} \chi_1(s,t) = SELECT & sum(Value) \\ FROM & CashBudget \\ WHERE & Section = s \\ AND & Type = t \end{array}$$

# Complexity results under SACs

- even if SACs are a restricted form of (general) aggregate constraints, results obtained for (general) aggregate constraints are still valid for SACs
- the repair existence problem
  - deciding whether there is a repair for a database violating a given set of SACs is NP-complete
- the minimal repair checking problem
  - deciding whether a repair is minimal in CoNP-complete
- the consistent query answer problem
  - deciding whether a query is true in every card-minimal repair is $\Delta_2^p[log\ n] - complete$

# Outline

- Repairing strategies
- DART architecture
- Aggregate constraints
- Steady aggregate constraints (SAC)
- Computing a card-minimal repair

# Repairing Module – MILP transformer

- Under SACs a card-minimal repair can be computed solving an MILP problem instance
  - SACs are translated into a system of inequalities $A\,Z \leq B$
    - $Z=[z_1,z_2,\ldots,z_N]$ is a vector of variables associated to database values $v_1,v_2,\ldots,v_N$ which are involved in a constraint

| Section | Subsection | Type | Value | |
|---|---|---|---|---|
| Receipts | beginning cash | *drv* | *20* | |
| Receipts | cash sales | *det* | *100* | $z_1$ |
| Receipts | receivables | *det* | *120* | $z_2$ |
| Receipts | total cash receipts | *aggr* | *250* | $z_3$ |
| Disbursements | payment of accounts | *det* | *120* | $z_4$ |
| Disbursements | capital expenditure | *det* | *0* | $z_5$ |
| Disbursements | long-term financing | *det* | *40* | $z_6$ |
| Disbursements | total disbursements | *aggr* | *160* | $z_7$ |
| Balance | net cash inflow | *drv* | *60* | |
| Balance | ending cash balance | *drv* | *80* | |

1) $\begin{cases} z_1 + z_2 = z_3 \\ z_4 + z_5 + z_6 = z_7 \end{cases}$

1) $CashBudget(s,\_,\_,\_) \implies \chi_1(s,det) - \chi_1(s,aggr) = 0$

# Repairing Module – MILP transformer

- Under SACs a card-minimal repair can be computed solving an MILP problem instance
  - SACs are translated into a system of inequalities $A\ Z \leq B$
    - $Z=[z_1,z_2,\ldots,z_N]$ is a vector of variables associated to database values $v_1,v_2,\ldots,v_N$ which are involved in a constraint

| Section | Subsection | Type | Value | |
|---|---|---|---|---|
| Receipts | beginning cash | *drv* | *20* | |
| Receipts | cash sales | *det* | *100* | → $z_1$ |
| Receipts | receivables | *det* | *120* | → $z_2$ |
| Receipts | total cash receipts | *aggr* | *250* | → $z_3$ |
| Disbursements | payment of accounts | *det* | *120* | $z_4$ |
| Disbursements | capital expenditure | *det* | *0* | $z_5$ |
| Disbursements | long-term financing | *det* | *40* | $z_6$ |
| Disbursements | total disbursements | *aggr* | *160* | $z_7$ |
| Balance | net cash inflow | *drv* | *60* | → $z_8$ |
| Balance | ending cash balance | *drv* | *80* | |

1) $$\begin{cases} z_1 + z_2 = z_3 \\ z_4 + z_5 + z_6 = z_7 \end{cases}$$

2) $$z_3 - z_7 = z_8$$

2) $$CashBudget(\_,\_,\_,\_) \implies \chi_2(net\ cash\ inflow) - [\chi_2(total\ cash\ receipts) - \chi_2(total\ disbursements)] = 0$$

# Repairing Module – MILP transformer

- Under SACs a card-minimal repair can be computed solving an MILP problem instance
  - SACs are translated into a system of inequalities $A\ Z \leq B$
    - $Z=[z_1,z_2,\ldots,z_N]$ is a vector of variables associated to database values $v_1,v_2,\ldots,v_N$ which are involved in a constraint

| **Section** | **Subsection** | **Type** | **Value** |
|---|---|---|---|
| Receipts | beginning cash | *drv* | *20* |
| Receipts | cash sales | *det* | *100* |
| Receipts | receivables | *det* | *120* |
| Receipts | total cash receipts | *aggr* | *250* |
| Disbursements | payment of accounts | *det* | *120* |
| Disbursements | capital expenditure | *det* | *0* |
| Disbursements | long-term financing | *det* | *40* |
| Disbursements | total disbursements | *aggr* | *160* |
| Balance | net cash inflow | *drv* | *60* |
| Balance | ending cash balance | *drv* | *80* |

$z_1$
$z_2$
$z_3$
$z_4$
$z_5$
$z_6$
$z_7$
$z_8$

$$\begin{cases} z_1 + z_2 = z_3 \\ z_4 + z_5 + z_6 = z_7 \\ z_3 - z_7 = z_8 \end{cases}$$

# Repairing Module – MILP transformer

- Under SACs a card-minimal repair can be computed solving an MILP problem instance
  - SACs are translated into a system of inequalities $A\,Z \leq B$
    - $Z=[z_1,z_2,\ldots,z_N]$ is a vector of variables associated to database values $v_1,v_2,\ldots,v_N$ which are involved in a constraint

| **Section** | **Subsection** | **Type** | **Value** |
|---|---|---|---|
| Receipts | beginning cash | *drv* | *20* |
| Receipts | cash sales | *det* | *100* |
| Receipts | receivables | *det* | *120* |
| Receipts | total cash receipts | *aggr* | *250* |
| Disbursements | payment of accounts | *det* | *120* |
| Disbursements | capital expenditure | *det* | *0* |
| Disbursements | long-term financing | *det* | *40* |
| Disbursements | total disbursements | *aggr* | *160* |
| Balance | net cash inflow | *drv* | *60* |
| Balance | ending cash balance | *drv* | *80* |

$z_1=130$

$z_2=120$

$z_3=250$

$z_4=120$

$z_5=0$

$z_6=40$

$z_7=160$

$z_8=90$

$$\begin{cases} z_1 + z_2 = z_3 \\ z_4 + z_5 + z_6 = z_7 \\ z_3 - z_7 = z_8 \end{cases}$$

each solution corresponds to a (possible not minimal) repair

# Repairing Module – MILP transformer

- In order to decide whether a solution corresponds to a card-minimal repair
  - we define a variable $y_i = z_i - v_i$

$$\begin{cases} z_1 + z_2 = z_3 \\ z_4 + z_5 + z_6 = z_7 \\ z_3 - z_7 = z_8 \end{cases}$$

| Section | Subsection | Type | Value | |
|---|---|---|---|---|
| Receipts | beginning cash | *drv* | *20* | |
| Receipts | cash sales | *det* | *100* | → $z_1$ |
| Receipts | receivables | *det* | *120* | → $z_2$ |
| Receipts | total cash receipts | *aggr* | *250* | → $z_3$ |
| Disbursements | payment of accounts | *det* | *120* | $z_4$ |
| Disbursements | capital expenditure | *det* | *0* | $z_5$ |
| Disbursements | long-term financing | *det* | *40* | $z_6$ |
| Disbursements | total disbursements | *aggr* | *160* | $z_7$ |
| Balance | net cash inflow | *drv* | *60* | → $z_8$ |
| Balance | ending cash balance | *drv* | *80* | |

# Repairing Module – MILP transformer

- In order to decide whether a solution corresponds to a card-minimal repair
  - we define a variable $y_i = z_i - v_i$

| Section | Subsection | Type | Value | |
|---|---|---|---|---|
| Receipts | beginning cash | *drv* | *20* | |
| Receipts | cash sales | *det* | *100* | → $z_1$ |
| Receipts | receivables | *det* | *120* | → $z_2$ |
| Receipts | total cash receipts | *aggr* | *250* | → $z_3$ |
| Disbursements | payment of accounts | *det* | *120* | $z_4$ |
| Disbursements | capital expenditure | *det* | *0* | $z_5$ |
| Disbursements | long-term financing | *det* | *40* | $z_6$ |
| Disbursements | total disbursements | *aggr* | *160* | $z_7$ |
| Balance | net cash inflow | *drv* | *60* | → $z_8$ |
| Balance | ending cash balance | *drv* | *80* | |

$$
\begin{cases}
z_1 + z_2 = z_3 \\
z_4 + z_5 + z_6 = z_7 \\
z_3 - z_7 = z_8 \\
y_1 = z_1 - 100 \\
y_2 = z_2 - 120 \\
y_3 = z_3 - 250 \\
y_4 = z_4 - 120 \\
y_5 = z_5 - 0 \\
y_6 = z_6 - 40 \\
y_7 = z_7 - 160 \\
y_8 = z_8 - 60
\end{cases}
$$

# Repairing Module – MILP transformer

- In order to decide whether a solution corresponds to a card-minimal repair
  - we define a variable $y_i = z_i - v_i$

| Section | Subsection | Type | Value |
|---|---|---|---|
| Receipts | beginning cash | *drv* | *20* |
| Receipts | cash sales | *det* | *100* |
| Receipts | receivables | *det* | *120* |
| Receipts | total cash receipts | *aggr* | *250* |
| Disbursements | payment of accounts | *det* | *120* |
| Disbursements | capital expenditure | *det* | *0* |
| Disbursements | long-term financing | *det* | *40* |
| Disbursements | total disbursements | *aggr* | *160* |
| Balance | net cash inflow | *drv* | *60* |
| Balance | ending cash balance | *drv* | *80* |

$z_1=130$
$z_2=120$
$z_3=250$
$z_4=120$
$z_5=0$
$z_6=40$
$z_7=160$
$z_8=90$

$$
\begin{cases}
z_1 + z_2 = z_3 \\
z_4 + z_5 + z_6 = z_7 \\
z_3 - z_7 = z_8 \\
y_1 = z_1 - 100 \\
y_2 = z_2 - 120 \\
y_3 = z_3 - 250 \\
y_4 = z_4 - 120 \\
y_5 = z_5 - 0 \\
y_6 = z_6 - 40 \\
y_7 = z_7 - 160 \\
y_8 = z_8 - 60
\end{cases}
$$

# Repairing Module – MILP transformer

- In order to decide whether a solution corresponds to a card-minimal repair
  - we define a variable $y_i = z_i - v_i$

| Section | Subsection | Type | Value |
|---|---|---|---|
| Receipts | beginning cash | *drv* | *20* |
| Receipts | cash sales | *det* | *100* |
| Receipts | receivables | *det* | *120* |
| Receipts | total cash receipts | *aggr* | *250* |
| Disbursements | payment of accounts | *det* | *120* |
| Disbursements | capital expenditure | *det* | *0* |
| Disbursements | long-term financing | *det* | *40* |
| Disbursements | total disbursements | *aggr* | *160* |
| Balance | net cash inflow | *drv* | *60* |
| Balance | ending cash balance | *drv* | *80* |

$z_1=130$ → $y_1=30$

$z_2=120$ → $y_2=0$

$z_3=250$ → $y_3=0$

$z_4=120$ → $y_4=0$

$z_5=0$ → $y_5=0$

$z_6=40$ → $y_6=0$

$z_7=160$ → $y_7=0$

$z_8=90$ → $y_8=30$

$$\begin{cases} z_1 + z_2 = z_3 \\ z_4 + z_5 + z_6 = z_7 \\ z_3 - z_7 = z_8 \\ y_1 = z_1 - 100 \\ y_2 = z_2 - 120 \\ y_3 = z_3 - 250 \\ y_4 = z_4 - 120 \\ y_5 = z_5 - 0 \\ y_6 = z_6 - 40 \\ y_7 = z_7 - 160 \\ y_8 = z_8 - 60 \end{cases}$$

$y_i \neq 0$ → atomic updated on database value $v_i$

# Repairing Module – MILP transformer

- In order to decide whether a solution corresponds to a card-minimal repair
  - we define a variable $y_i = z_i - v_i$

| Section | Subsection | Type | Value |
|---|---|---|---|
| Receipts | beginning cash | *drv* | *20* |
| Receipts | cash sales | *det* | *100* |
| Receipts | receivables | *det* | *120* |
| Receipts | total cash receipts | *aggr* | *250* |
| Disbursements | payment of accounts | *det* | *120* |
| Disbursements | capital expenditure | *det* | *0* |
| Disbursements | long-term financing | *det* | *40* |
| Disbursements | total disbursements | *aggr* | *160* |
| Balance | net cash inflow | *drv* | *60* |
| Balance | ending cash balance | *drv* | *80* |

$z_1=130$ $y_1=30$
$z_2=120$ $y_2=0$
$z_3=250$ $y_3=0$
$z_4=120$ $y_4=0$
$z_5=0$ $y_5=0$
$z_6=40$ $y_6=0$
$z_7=160$ $y_7=0$
$z_8=90$ $y_8=30$

$$\begin{cases} z_1 + z_2 = z_3 \\ z_4 + z_5 + z_6 = z_7 \\ z_3 - z_7 = z_8 \\ y_1 = z_1 - 100 \\ y_2 = z_2 - 120 \\ y_3 = z_3 - 250 \\ y_4 = z_4 - 120 \\ y_5 = z_5 - 0 \\ y_6 = z_6 - 40 \\ y_7 = z_7 - 160 \\ y_8 = z_8 - 60 \end{cases}$$

  - we have to count the number of variables $y_i$ such that $y_i \neq 0$

# Repairing Module – MILP transformer

- In order to detect if a variable $z_i$ is assigned a value different $v_i$, a binary variable $\delta_i$ is defined
- we add the following constraints entailing that $y_i \neq 0 \Rightarrow \delta_i = 1$

$y_i \leq M\delta_i$ → $y_i > 0$ implies $\delta_i = 1$

$-M\delta_i \leq y_i$ → $y_i < 0$ implies $\delta_i = 1$

If a system of equalities has a solution, it has also one where each variable takes a value in [-M,M]

# Repairing Module – MILP transformer

- In order to detect if a variable $z_i$ is assigned (for each *M*-bounded solution) a value different $v_i$, a binary variable $\delta_i$ is defined
- we add the following constraints entailing that $y_i \neq 0 \Rightarrow \delta_i = 1$

$y_i \leq M\delta_i$ → $y_i > 0$ implies $\delta_i = 1$

$-M\delta_i \leq y_i$ → $y_i < 0$ implies $\delta_i = 1$

$$
\begin{cases}
z_1 + z_2 = z_3 \\
z_4 + z_5 + z_6 = z_7 \\
z_3 - z_7 = z_8 \\
y_1 = z_1 - 100 \\
y_2 = z_2 - 120 \\
y_3 = z_3 - 250 \\
y_4 = z_4 - 120 \\
y_5 = z_5 - 0 \\
y_6 = z_6 - 40 \\
y_7 = z_7 - 160 \\
y_8 = z_8 - 60
\end{cases}
\qquad
\begin{cases}
y_1 \leq M\delta_1 \\
-M\delta_1 \leq y_1 \\
y_2 \leq M\delta_2 \\
-M\delta_2 \leq y_2 \\
\ldots \\
\ldots \\
y_8 \leq M\delta_8 \\
-M\delta_8 \leq y_8
\end{cases}
$$

| | |
|---|---|
| $z_1=130$ | $y_1=30$ |
| $z_2=120$ | $y_2=0$ |
| $z_3=250$ | $y_3=0$ |
| $z_4=120$ | $y_4=0$ |
| $z_5=0$ | $y_5=0$ |
| $z_6=40$ | $y_6=0$ |
| $z_7=160$ | $y_7=0$ |
| $z_8=90$ | $y_8=30$ |

→ $\delta_1 = 1$

# Repairing Module – MILP transformer

- In order to detect if a variable $z_i$ is assigned (for each *M*-bounded solution) a value different $v_i$, a binary variable $\delta_i$ is defined
- we add the following constraints entailing that $y_i \neq 0 \Rightarrow \delta_i = 1$

$y_i \leq M\delta_i$ → $y_i > 0$ implies $\delta_i = 1$

$-M\delta_i \leq y_i$ → $y_i < 0$ implies $\delta_i = 1$

$$\begin{cases} z_1 + z_2 = z_3 \\ z_4 + z_5 + z_6 = z_7 \\ z_3 - z_7 = z_8 \\ y_1 = z_1 - 100 \\ y_2 = z_2 - 120 \\ y_3 = z_3 - 250 \\ y_4 = z_4 - 120 \\ y_5 = z_5 - 0 \\ y_6 = z_6 - 40 \\ y_7 = z_7 - 160 \\ y_8 = z_8 - 60 \end{cases}$$

$$\begin{cases} y_1 \leq M\delta_1 \\ -M\delta_1 \leq y_1 \\ y_2 \leq M\delta_2 \\ -M\delta_2 \leq y_2 \\ \ldots \\ \ldots \\ y_8 \leq M\delta_8 \\ -M\delta_8 \leq y_8 \end{cases}$$

| | |
|---|---|
| $z_1=130$ | $y_1=30$ |
| $z_2=120$ | $y_2=0$ |
| $z_3=250$ | $y_3=0$ |
| $z_4=120$ | $y_4=0$ |
| $z_5=0$ | $y_5=0$ |
| $z_6=40$ | $y_6=0$ |
| $z_7=160$ | $y_7=0$ |
| $z_8=90$ | $y_8=30$ |

$y_i=0$ entails that

either $\delta_i=1$ or $\delta_i=0$

# Repairing Module – MILP transformer

- In order to consider solutions where each $\delta_i=0$ if $y_i=0$, we minimize the sum of values assigned to binary variables $\delta_i$

$$\min\ \delta_1+\delta_2+\ldots+\delta_8$$

$$\begin{cases} z_1 + z_2 = z_3 \\ z_4 + z_5 + z_6 = z_7 \\ z_3 - z_7 = z_8 \\ y_1 = z_1 - 100 \\ \quad \ldots \\ y_8 = z_8 - 60 \\ y_1 \le M\delta_1 \\ -M\delta_1 \le y_1 \\ \quad \ldots \\ y_8 \le M\delta_8 \\ -M\delta_8 \le y_8 \end{cases}$$

1. any solution corresponds to an *M*-bounded repair having minimum cardinality w.r.t. all *M*-bounded repairs
2. It can be shown that if a repair exists then there is a card-minimal repair that is *M*-bounded

↓

any solution corresponds to a card-minimal repair

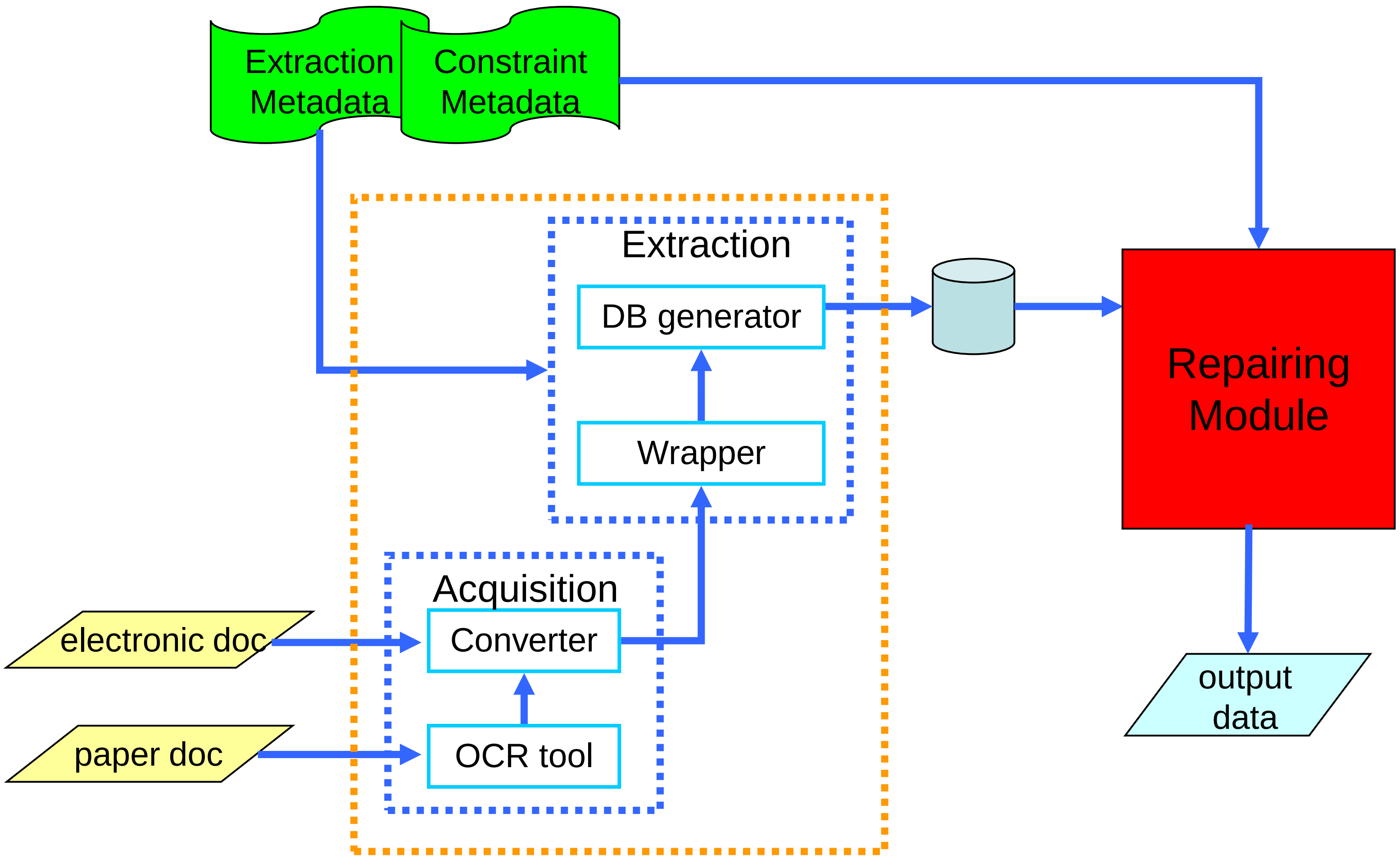
# Conclusions and future work

- An architecture providing robust data acquisition facilities has been proposed
- A restricted, but useful in many real-life scenario, class of aggregate constraints has been located
- An approach for computing a card-minimal repair in presence of SACs has been provided
  - standard techniques addressing MILP problem can be re-used for computing a repair
- Experimental evaluation of the system effectiveness on large data sets (working with real databases) will be accomplished

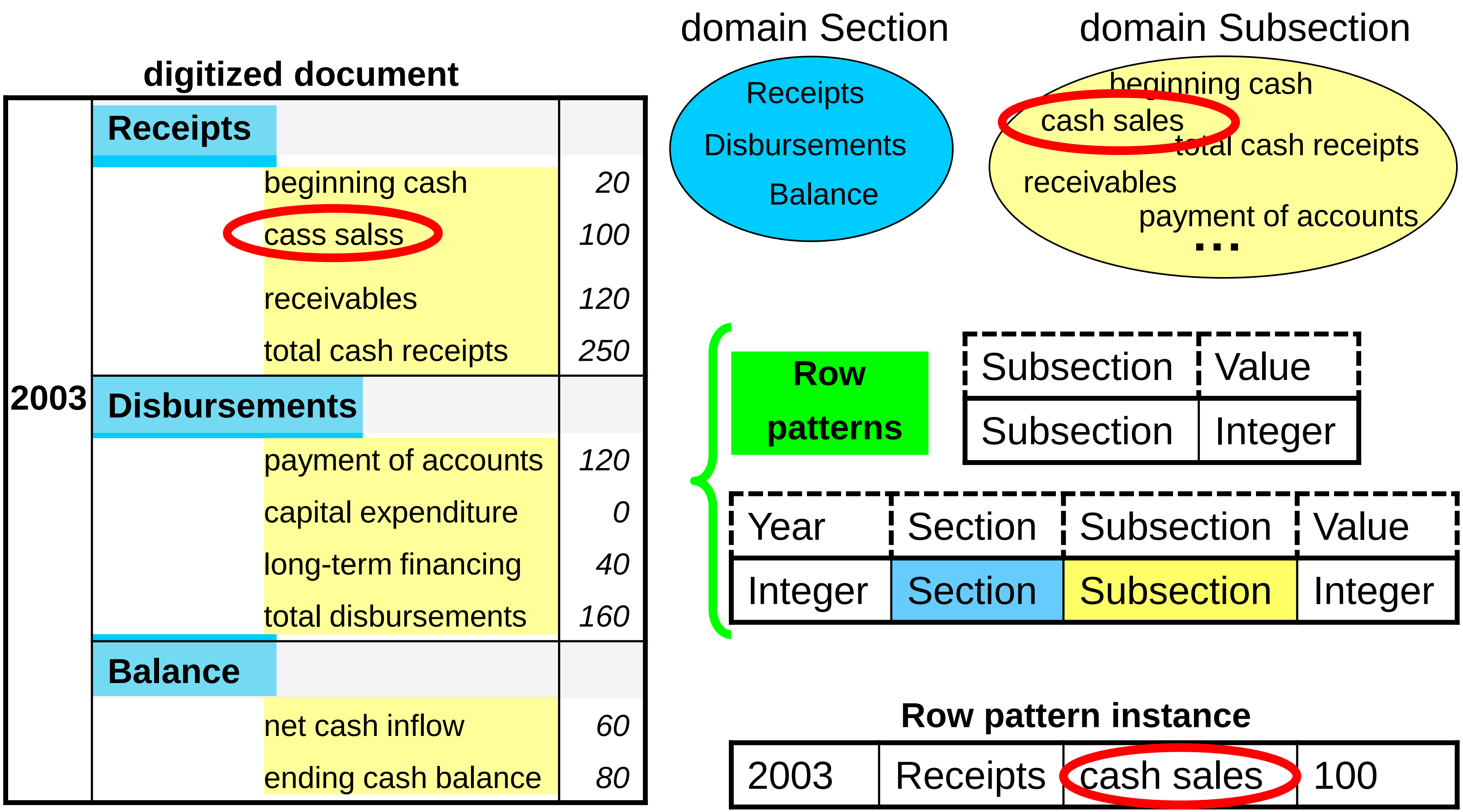
Thank you!

...*any questions?*

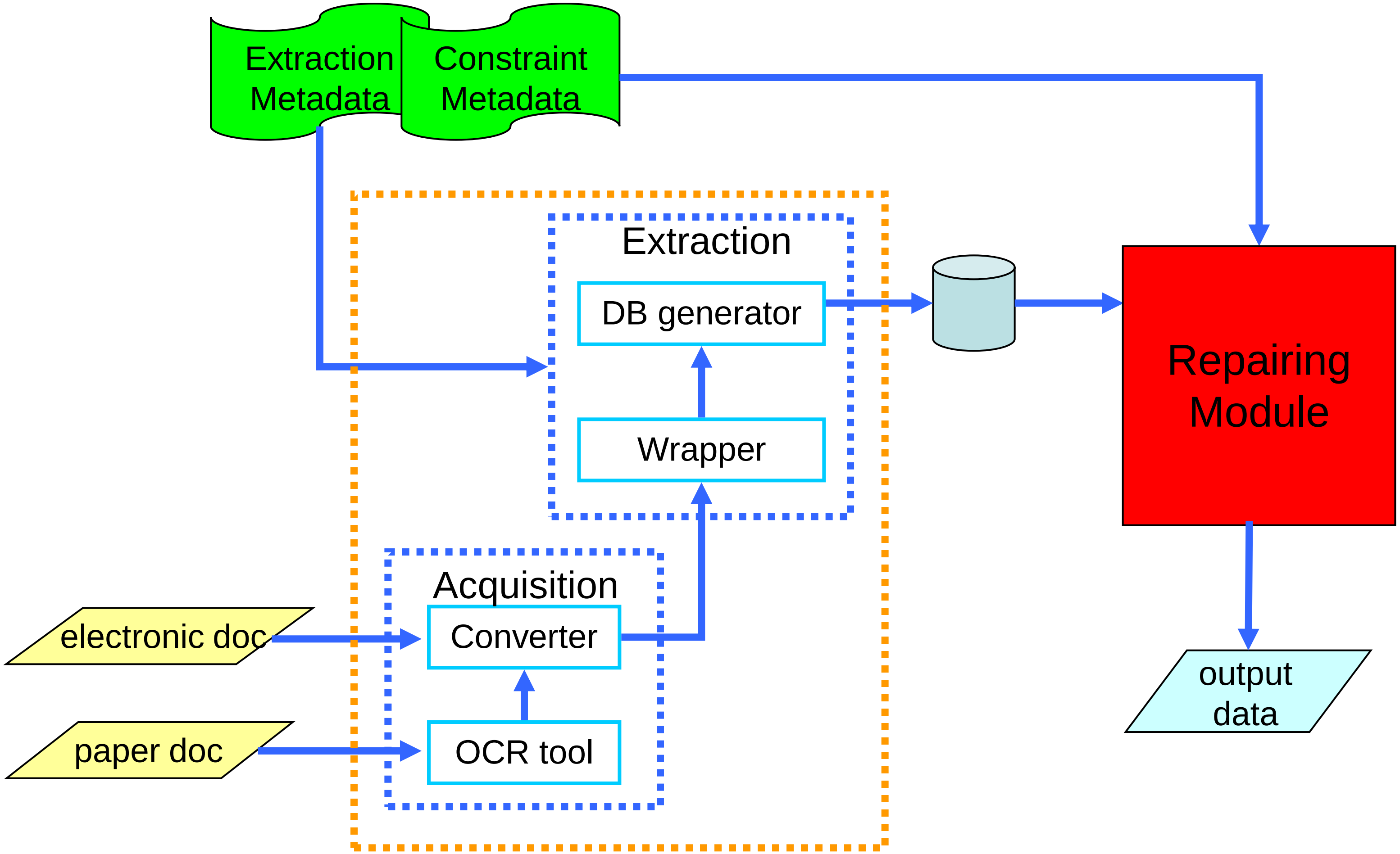
# DART architecture - Acquisition and Extraction Module



# Data Extraction Sub-Module - Wrapper



# DART architecture - Acquisition and Extraction Module



# Data Extraction Sub-Module – DB generator

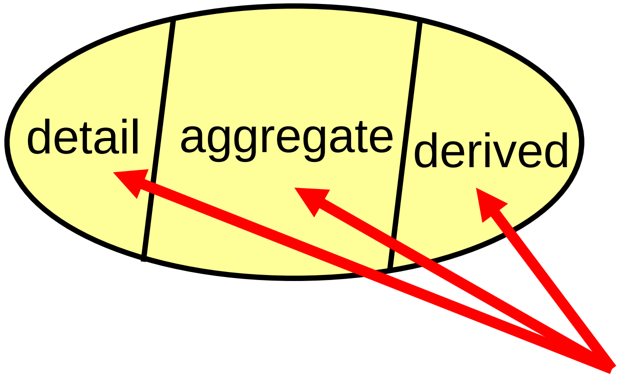
**Row pattern**

| Year | Section | Subsection | Value |
|---|---|---|---|
| Integer | Section | Subsection | Integer |

**Row pattern instances**

| | | | |
|---|---|---|---|
| 2003 | Receipts | beginning cash | 20 |
| 2003 | Receipts | cash sales | 100 |
| 2003 | Receipts | receivables | 120 |
| 2003 | Receipts | total cash receipts | 250 |

Subsection




CashBudget(Year,Section,Subsection,Type,Value)

CashBudget

| Year | Section | Subsection | Type | Value |
|---|---|---|---|---|
| 2003 | Receipts | beginning cash | *drv* | *20* |
| 2003 | Receipts | cash sales | *det* | *100* |
| 2003 | Receipts | receivables | *det* | *120* |
| 2003 | Receipts | total cash receipts | *aggr* | *250* |
| ... | ... | ... | ... | ... |