Mapper
An Efficient Data Transformation Operator

Paulo Carreira
University of Lisbon
## Context

Source: **IMPCRED**

<table>
<thead>
<tr>
<th>CREDTID</th>
<th>MERCH</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT546083</td>
<td>8 PCS LARGE CONTAINER MODULE X067</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>100 PCS COTTON CORDUROY 8 500 MTS XS03 208</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>63 PCS POLYESTER TUBE W987</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>34 PCS ARTIFICIAL FIBRE</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>3 PCS SPANDEX 2 400 MTS W15719</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>83 PCS POLYESTER TUBE W989</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>15 PCS ARTIFICIAL FIBRE 3 500 MTS NEW CHECK DESIGN</td>
<td>...</td>
</tr>
<tr>
<td>RT546084</td>
<td>4800 PCS IC CAN TRANSCEIVER 3.3V 8-SOIC SN65HVD232D</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>2400 PCS PCA9532BS-T 16-bit I2C LED DIMMER</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORDRID</th>
<th>QTY</th>
<th>DESCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>546083</td>
<td>8</td>
<td>LARGE CONTAINER MODULE X067</td>
</tr>
<tr>
<td>546083</td>
<td>100</td>
<td>COTTON CORDUROY 8 500 MTS XS03 208</td>
</tr>
<tr>
<td>546083</td>
<td>63</td>
<td>POLYESTER TUBE W987</td>
</tr>
<tr>
<td>546083</td>
<td>34</td>
<td>ARTIFICIAL FIBRE</td>
</tr>
<tr>
<td>546083</td>
<td>3</td>
<td>SPANDEX 2 400 MTS W15719</td>
</tr>
<tr>
<td>546083</td>
<td>83</td>
<td>POLYESTER TUBE W989</td>
</tr>
<tr>
<td>546083</td>
<td>15</td>
<td>ARTIFICIAL FIBRE 3 500 MTS NEW CHECK DESIGN</td>
</tr>
<tr>
<td>546084</td>
<td>4800</td>
<td>IC CAN TRANSCEIVER 3.3V 8-SOIC SN65HVD232D</td>
</tr>
<tr>
<td>546084</td>
<td>2400</td>
<td>PCA9532BS-T 16-bit I2C LED DIMMER</td>
</tr>
</tbody>
</table>
## State of the art

<table>
<thead>
<tr>
<th>General Purpose Language</th>
<th>ETL Tool</th>
<th>RDBMS</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Union</td>
<td>Unpivot</td>
</tr>
<tr>
<td>Declarativeness</td>
<td>-</td>
<td>+/-</td>
<td>+</td>
</tr>
<tr>
<td>Optimizability</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Expressiveness</td>
<td>+</td>
<td>+/-</td>
<td>-</td>
</tr>
</tbody>
</table>
Main Contributions

1. Extending of Relational Algebra
   - **Data Mapper**: An operator for expressing One-to-many data transformations

2. Logical optimization rules for the mapper operator, e.g.,
   - Pushdown of projections
   - Pushdown of selections

3. Physical execution algorithms for the mapper operator
   - Naïve algorithm
   - Shortcircuiting algorithm
   - Cache-based algorithm
The mapper operator

\[ \mu_F(s) = \bigcup_{u \in s} f_{A_1}(u) \times \ldots \times f_{A_k}(u) \]

**Source schema**

IMPCRED(CREDID, MERCHANDISE)

**Target schema**

MERCHDETL(ORDERID, QTY, DESCR)

**Source data**

- CREDID: 546084
- MERCHANDISE: 4800 PCS IC CAN TRANSCEIVER 3.3V 8-SOIC SN65HVD232D
- 2400 PCS PCA9532BS-T 16-bit I2C LED DIMMER

**Target data**

- ORDERID: 546084
- QTY: 4800
- DESCR: IC CAN TRANSCEIVER 3.3V 8-SOIC SN65HVD232D
- ORDERID: 546084
- QTY: 2400
- DESCR: PCA9532BS-T 16-bit I2C LED DIMMER
Algebraic Optimization

- Distributing mappers over Cartesian products
- Distributing mappers over unions
- Pushdown of selections
- Pushdown of projections
- Pushing selections to mapper functions
- Introduction of projections

```
\[ \pi_Z \]
1

\[ \mu_F \]

\[ \sigma_{ORDRID < 500000 \land \text{GARTYP} \neq 'R'} \]

\[ \mu_{cid, extr, gt} \]

\[ \sigma_{C_B} \]

\[ \sigma_{C_A} \]

\[ \mu_F \]

\[ \mu_F(s) \]

\[ \text{IMPORTCRED} \]

\[ \times \text{ACCTNO=CREDAC} \]

\[ \text{ACCOUNTS} \]

\[ \times \text{COLLATCLT=CLTID} \]

\[ \text{COLLATERALS} \]

\[ R' \]
Physical Algorithms

Cache-based

Naïve Duplicate input values
Functions returning empty sets

Naïve Shortcircuiting

Physical Algorithms

Expensive Functions

Cheap Functions

Input relation

Cache of Function Results

LRU

LRU

LUR

XLUR

Duplicate input values

LRU stack
Utility list
Multiple LRU Stacks

Output relation

Approximate Utility

Cache

Next victim?

Replacement Policies

Most Recently Used
Least Recently Used
Most Useful
Least Useful
LF
LE
LRU

MF
MRU
ME

Multiple LRU Stacks

Approximate Utility

Least Useful

Most Useful

Least Recently Used

Most Recently Used

LF
LE
LRU

MF
MRU
ME

Multiple LRU Stacks
Experimental Validation

1. **Comparison of RDBMSs with mappers**
   - Mappers are 1.7x – 3.6x faster than the best RDBMS solution

2. **Gain of logical optimizations**
   - Gains of 1.3x – 5.5x for pushing selections

3. **Performance of physical algorithms**
   - Shortcircuiting algorithm: 50x faster than the Naïve (at ~1ms/call)
   - Cache-based algorithm (XLUR)
     - Interesting savings: 50% duplicates = 1.4x faster than the Naïve
     - Lightweight: virtually the same overhead as LRU
     - Successful at performing cost-based decisions: outperforms LRU for low CHRs
Conclusions

- New unary operator extension to Relational Algebra: Data Mapper

- Addresses: One-to-many data transformations
  - Declarative
  - Optimizable
  - Expressive

- Consequences:
  - Broadens the span of application of RDBMSs
  - Significant improvement for ETL tools (Data Fusion tool)
Selected Publications


Reserve Slides
Future work

- Study mapper functions
  - Study the properties of mapper functions
  - Develop a taxonomy of mapper functions
  - Propose a library of useful mapper functions

- Enhance the physical algorithms

- Incorporate the mapper operator on an Open Source RDBMS
# Extracting rows from unstructured data

**Source:** CITEDATA

<table>
<thead>
<tr>
<th>NAME</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jorge Vieira, Jorge Bernardino, Henrique Madeira</td>
<td>Efficient Compression of Text Attributes of Data Warehouse Dimensions</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Target:** EVENTS

<table>
<thead>
<tr>
<th>NAME</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Periklis Andritsos</td>
<td>Schema Management</td>
</tr>
<tr>
<td>Ronald Fagin</td>
<td>Schema Management</td>
</tr>
<tr>
<td>Ariel Fuxman</td>
<td>Schema Management</td>
</tr>
<tr>
<td>Laura M. Haas</td>
<td>Schema Management</td>
</tr>
<tr>
<td>Mauricio A. Hernández</td>
<td>Schema Management</td>
</tr>
<tr>
<td>C. T. Howard Ho</td>
<td>Schema Management</td>
</tr>
<tr>
<td>Anastasios Kementsietsidis</td>
<td>Schema Management</td>
</tr>
<tr>
<td>Renée J. Miller</td>
<td>Schema Management</td>
</tr>
<tr>
<td>Felix Naumann</td>
<td>Schema Management</td>
</tr>
<tr>
<td>Lucian Popa</td>
<td>Schema Management</td>
</tr>
<tr>
<td>Yannis Velegrakis</td>
<td>Schema Management</td>
</tr>
<tr>
<td>Charlotte Vilarem</td>
<td>Schema Management</td>
</tr>
<tr>
<td>Ling-Ling Yan</td>
<td>Schema Management</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Example: Converting lines into columns

**Source:** LOANEVT

<table>
<thead>
<tr>
<th>LOANO</th>
<th>EVTYP</th>
<th>CAPTL</th>
<th>TAX</th>
<th>EXPNS</th>
<th>BONUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>OPEN</td>
<td>0.0</td>
<td>0.19</td>
<td>0.28</td>
<td>0.1</td>
</tr>
<tr>
<td>1234</td>
<td>PAY</td>
<td>1000.0</td>
<td>0.28</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1234</td>
<td>PAY</td>
<td>1250.0</td>
<td>0.30</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1234</td>
<td>EARLY</td>
<td>550.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1234</td>
<td>FULL</td>
<td>5000.0</td>
<td>1.1</td>
<td>5.0</td>
<td>3.0</td>
</tr>
<tr>
<td>1234</td>
<td>CLOSED</td>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**Target:** EVENTS

<table>
<thead>
<tr>
<th>LOANO</th>
<th>EVTYP</th>
<th>AMTYP</th>
<th>AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>OPEN</td>
<td>TAX</td>
<td>0.19</td>
</tr>
<tr>
<td>1234</td>
<td>OPEN</td>
<td>EXPNS</td>
<td>0.28</td>
</tr>
<tr>
<td>1234</td>
<td>OPEN</td>
<td>BONUS</td>
<td>0.1</td>
</tr>
<tr>
<td>1234</td>
<td>PAY</td>
<td>CAPTL</td>
<td>1000.0</td>
</tr>
<tr>
<td>1234</td>
<td>PAY</td>
<td>TAX</td>
<td>0.28</td>
</tr>
<tr>
<td>1234</td>
<td>PAY</td>
<td>CAPTL</td>
<td>1250.0</td>
</tr>
<tr>
<td>1234</td>
<td>PAY</td>
<td>TAX</td>
<td>0.30</td>
</tr>
<tr>
<td>1234</td>
<td>EARLY</td>
<td>CAPTL</td>
<td>550.0</td>
</tr>
<tr>
<td>1234</td>
<td>FULL</td>
<td>CAPTL</td>
<td>5000.0</td>
</tr>
<tr>
<td>1234</td>
<td>FULL</td>
<td>TAX</td>
<td>1.1</td>
</tr>
<tr>
<td>1234</td>
<td>FULL</td>
<td>EXPNS</td>
<td>5.0</td>
</tr>
<tr>
<td>1234</td>
<td>FULL</td>
<td>BONUS</td>
<td>3.0</td>
</tr>
<tr>
<td>1234</td>
<td>CLOSED</td>
<td>EXPNS</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Implementations of one-to-many data transformations

<table>
<thead>
<tr>
<th></th>
<th>Bounded</th>
<th>Unbounded</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RA/SQL</td>
<td>PSM</td>
</tr>
<tr>
<td>DBX</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>OEX</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>
## Extensions to Relational Algebra

<table>
<thead>
<tr>
<th>Type of data transformation</th>
<th>Unary Extension to RA</th>
<th>Type of function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Many-to-one</td>
<td>Aggregation</td>
<td>Aggregate Function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set $\rightarrow$ Value</td>
</tr>
<tr>
<td>One-to-one</td>
<td>Extended Projection</td>
<td>Scalar Function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value $\rightarrow$ Value</td>
</tr>
<tr>
<td>One-to-many</td>
<td>Mapper</td>
<td>Mapper Function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value $\rightarrow$ Set</td>
</tr>
</tbody>
</table>
Mapper example

1: select
2:   lpad(tostr(ACCT), 4, '0') as ACCTNO,
3:   map AMOUNT, SEQNO
4: begin
5:   var rem_amnt: numeric
6:   var seq_no: integer  = 0
7:   rem_amnt = AMT
8: loop while rem_amnt > 100 do
9:   rem_amnt = rem_amnt - 100
10:  seq_no = seq_no + 1
11:  insert rem_amnt, seq_no
12: end loop
13: if rem_amnt > 0 then
14:  insert rem_amnt, seq_no + 1
15: end if
16: end
17: from LOANS
Simplified SELECT syntax for mappers

\[
\text{select}
\]

\[
\text{select} \quad \text{mapperfunc} \quad \text{from} \quad \text{table}
\]

Syntactic details for mapper functions

\[
\text{mapperfunc}
\]

\[
\text{colname} \quad \text{as} \quad \text{colname}
\]

\[
\text{expr} \quad \text{as} \quad \text{colname}
\]

\[
\text{map} \quad \text{func} \quad ( \quad \text{colname} \quad )
\]

\[
\text{map} \quad \text{outputcols} \quad \text{as} \quad \text{outputcols}
\]

\[
\text{col}
\]
Example: Query with the mapper operator

1: select map acct(ACCT) as ACCTNO,
2:      map amt(AM) as AMOUNT
3: from LOANS, ACCOUNTS
4: where ACCOUNTS.ACCTN = LOANS.ACCT
5:   and ACCOUNTS.STATUS = 'O'
6:   and AMOUNT < 50
```sql
1: insert into EVENTS (LOANNO, EVTYP, AMTYP, AMT)
2:    select LOANNO, EVTYP, 'CAPTL' as AMTYP, CAPTL
3:    from LOANEVT
4:    where CAPTL > 0
5:    union all
6:    select LOANNO, EVTYP, 'TAX' as AMTYP, TAX
7:    from LOANEVT
8:    where TAX > 0
9:    union all
10:   select LOANNO, EVTYP, 'EXPNS' as AMTYP, EXPNS
11:    from LOANEVT
12:    where EXPNS > 0
13:   union all
14:   select LOANNO, EVTYP, 'BONUS' as AMTYP, BONUS
15:   from LOANEVT
16:   where BONUS > 0;
```
Recursive Query

1: with repayments(digits(ACCTNO), AMOUNT, SEQNO, REMAMNT) as
2:   (select ACCT,
3:       case when base.AM < 100 then base.AM
4:       else 100 end,
5:       1,
6:       case when base.AM < 100 then 0
7:       else base.AM - 100 end
8:       from LOANS as base
9:       union all
10:      select ACCTNO,
11:      case when step.REMAMNT < 100 then
12:      step.REMAMNT
13:      else 100 end,
14:      SEQNO + 1,
15:      case when step.REMAMNT < 100 then 0
16:      else step.REMAMNT - 100 end,
17:     from repayments as step
18:     where step.REMAMNT > 0)
19: select ACCTNO, SEQNO, AMOUNT
20: from repayments as PAYMENTS
Table function

An iterator that scans the input relation once

With a nested while loop that inserts pipes multiple several output tuples

1: create function LOANSTOPAYMENTS
2: return PAYMENTS_TABLE_TYPE pipelined is
3: ACCTVALUE LOANS.ACCT%TYPE;
4: AMVALUE LOANS.AMV%TYPE;
5: REMAMNT INT;
6: SEQNUM INT;
7: cursor LOANS is
8: select * from LOANS;
9: begin
10:  open LOANS;
11:  loop
12:   fetch LOANS into ACCTVALUE, AMVALUE;
13:   REMAMNT := AMVALUE;
14:   SEQNUM := 1;
15:   while REMAMNT > 100
16:     loop
17:       pipe row(PAYMENTS_ROW_TYPE(
18:         LPAD(ACCTVALUE, 4, '0'), 100.00, SEQNUM));
19:       REMAMNT := REMAMNT - 100;
20:       SEQNUM := SEQNUM + 1;
21:     end loop
22:     if REMAMNT > 0 then
23:       pipe row(PAYMENTS_ROW_TYPE(
24:         values (LPAD(ACCTVALUE, 4, '0'),
25:         REMAMNT, SEQNUM));
26:     end if
27:  end loop
28: end LOANSTOPAYMENTS
Algebraic rewriting rules

Pushdown of projections
\[ \pi_Z(\mu_F(s)) = \pi_Z(\mu_F'(s)) \]

Introduction of projections
\[ \mu_F(s) = \mu_F(\pi_N(s)) \]

Pushdown of selections
\[ \sigma_{C_B}(\mu_F(s)) = \mu_F(\sigma_{C[B_1,\ldots,B_m\leftarrow F|B_1,\ldots,F|B_m]}(s)) \]

Pushing selections to mapper functions
\[ \sigma_{C_{A_i}}(\mu_F(s)) = \mu_F(\sigma_{C_{A_i}}\circ f_{A_i} \cup \{f_{A_i}\})(s) \]

Distributing mappers over unions
\[ \mu_F(r \cup s) = \mu_F(r) \cup \mu_F(s) \]

Distributing mappers over Cartesian products
\[ \mu_F(s \times r) = \mu_{FS}(s) \times \mu_{FR}(r) \]
Statistics for computing the cost of One-to-many data transformations

- Given a mapper $\mu_F$
  1. Average selectivity of the mapper ($\alpha_F$)
     - Ratio of input tuples that are transformed
  2. Average fanout factor of the mapper ($O_F$)
     - Ratio of output tuples produced for each input tuple
  3. Average mapper function cost ($C_F$)
     - Cost of evaluating a mapper function
Estimates for cost-based plan selection

- Estimating the cardinality of a mapper expression
  \[ \text{card}(\mu_F(e)) = \text{card}(e) \cdot \alpha_F \cdot O_F \]
  - number of input tuples
  - input tuples that are not filtered
  - total number of output tuples
  - Fanout factor
  - Selectivity factor

- Estimating the CPU cost of a mapper expression
  \[ \text{cost}_{\text{CPU}}(\mu_F(e)) = \text{card}(e) \cdot (k \cdot O_F + C_F) \]
  - number of input tuples
  - CPU cost of the cartesian product operation
  - CPU cost of evaluating all the mapper functions
  - per-tuple CPU cost
  - total CPU cost
Cache-based evaluation 1/2

Cache of Function Results

LRU stack

Utility list

Input relation

... | ... | ...

Output relation

... | ... | ...

Cost

Freq

Recency

Utility

Next victim

LRU

LUR

O(m.log(m))
Cache-based evaluation 2/2

Input relation

Cache of Function Results

LRU stacks

Approximate Utility

Output relation

Selection of the Next Victim

XLRU
An utility metric based on statistical inference

Which entry to replace?

Utility $u_t(e)$ based on:
- Number of past references $n_h$
- Cost $c$
- Avg frequency $\theta$
- Time to last $k = (t_0 - t_1)$

Evolution of $(1-\theta)^k$
Experimental Setup

- Workload
  - **LOANEVT** and **LOANS** synthetic input relations (on raw devices)
  - Relations sizes: 100K, 500K, 1M, 2.5M, 5M (in tuples)
Settings checklist

- Same configuration for SGBDs
  - Same block size
  - Same multi-block read count
  - Same size for Buffer pools
  - Same concurrency settings (DBRD and DBWR processes)

- Align the configuration of the SGBDS with Java
  - Same physical conditions for I/O
  - Same record-size
  - Same number of records per page
  - Logging disabled
  - Asynchronous I/O disabled
  - Parallel query execution disabled
## Resource consumption

<table>
<thead>
<tr>
<th></th>
<th>Union</th>
<th>Unpivot</th>
<th>RQ</th>
<th>TF</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; fanout</td>
<td>more input</td>
<td>input once</td>
<td>more output</td>
<td>input once</td>
</tr>
<tr>
<td></td>
<td>more output</td>
<td>more output</td>
<td>more output</td>
<td>more output</td>
</tr>
<tr>
<td></td>
<td>bigger query</td>
<td>more output</td>
<td>more ‘temp’</td>
<td>more output</td>
</tr>
<tr>
<td>&gt; select</td>
<td>same input</td>
<td>same input</td>
<td>same input</td>
<td>same input</td>
</tr>
<tr>
<td></td>
<td>more output</td>
<td>more output</td>
<td>more ‘temp’</td>
<td>more output</td>
</tr>
</tbody>
</table>
Throughput of one-to-many transformations

- UN/SQL: ~50K
- PV/SQL: ~33K

Data transformation implementations:
Influence of selectivity

![Graph showing Influence of selectivity with lines for different selectivity levels and labels for similar and fastest degradation.]
Original v.s. optimized expression

Response Time [in seconds]

Predicate selectivity factor [in %]

mapper original
mapper optimized
table function original
table function optimized

3x faster

3x faster