Mapper
An Efficient Data Transformation Operator

Paulo Carreira
University of Lisbon

Context

Source: **IMPCRED**

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

Target: **MERCHDETL**

ORDRID	QTY	DESCR	…
546083	8	LARGE CONTAINER MODULE X067	…
546083	100	COTTON CORDUROY 8 500 MTS XS03 208	…
546083	63	POLYESTER TUBE W987	…
546083	34	ARTIFICIAL FIBRE	…
546083	3	SPANDEX 2 400 MTS W15719	…
546083	83	POLYESTER TUBE W989	…
546083	15	ARTIFICIAL FIBRE 3 500 MTS NEW CHECK DESIGN	…
546084	4800	IC CAN TRANSCEIVER 3.3V 8-SOIC SN65HVD232D	…
546084	2400	PCA9532BS-T 16-bit I2C LED DIMMER	…
…	…	…	…
…	…	…	…
…	…	…	…
…	…		…

State of the art

	General Purpose Language	ETL Tool	RDBMS				?
			Union	Unpivot	Recursive Queries	PSMs	
Declarativeness	-	+/-	+	+	+	+/-	+
Optimizability	-	-	+	+	+/-	-	+
Expressiveness	+	+/-	-	-	+	+	+

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)`

$s =$

CREDID	MERCHANDISE
PT546084	4800 PCS IC CAN TRANSCEIVER 3.3V 8-SOIC SN65HVD232D 2400 PCS PCA9532BS-T 16-bit I2C LED DIMMER
...	...

u

Target schema

`MERCHDETL(ORDERID, QTY, DESCR)`

ORDRID	QTY	DESCR
546084	4800	IC CAN TRANSCEIVER 3.3V 8-SOIC SN65HVD232D
546084	2400	PCA9532BS-T 16-bit I2C LED DIMMER
...	...	...

$$\mu_{cid_{\mathtt{ORDRID}},\, extr_{\mathtt{QTY,DESCR}}}(s)$$

$cid_{\mathtt{ORDRID}}(u)$

546084

t[ORDRID]

X

$extr_{\mathtt{QTY,\,DESCR}}(u)$

4800	IC CAN TRANSCEIVER 3.3V 8-SOIC SN65HVD232D
2400	PCA9532BS-T 16-bit I2C LED DIMMER

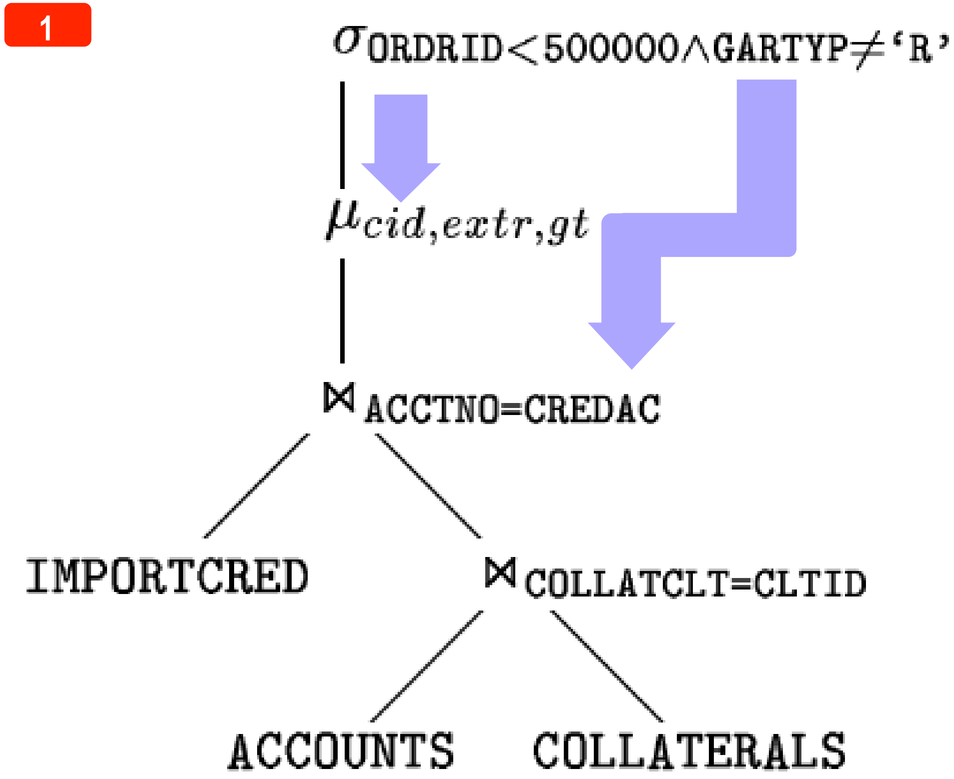
t[QTY, DESCR]

Algebraic Optimization

Pushdown o

$\pi_Z($

1



Introduction

$\mu_F($

Pushdown o

σ_{C_B}

$](s))$

Pushing sele

σ_{C_A}

R'

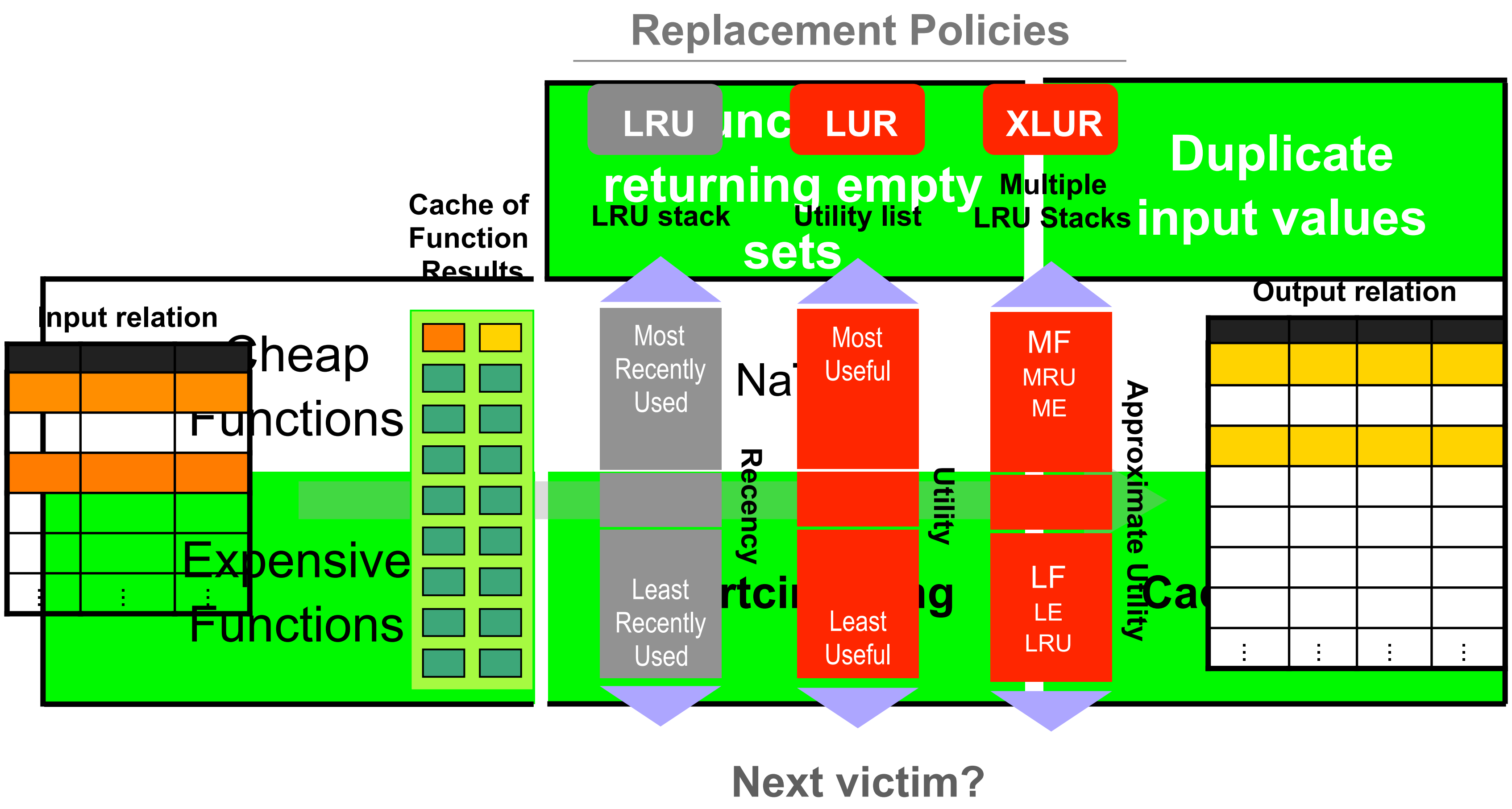
Distributing

$\mu_F($

Distributing

$\mu_F($

Physical Algorithms



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

- [Carreira et al. 2007] Carreira, P., Galhardas, H., Lopes, A. & Pereira, J., "*One-to-many Data Transformations through Data Mappers*", Data & Knowledge Engineering Journal (**DKE**), 62(3), 483–503, Elsevier-Science, **2007**.
- [Carreira et al. 2005] Carreira, P., Galhardas, H., Pereira, J., & Lopes, A., "*Data Mapper: An operator for expressing one-to-many data transformations*". 7th Int'l Conference on Data Warehousing and Knowledge Discovery, **DaWaK '05**, Vol. 3589 of LNCS, Springer-Verlag, **2005**.
- [Carreira et al. 2005b] Carreira, P., Galhardas, H., Lopes, A. & Pereira, J. (2005). "*Extending relational algebra to express one-to-many data transformations*". In 20th Brazillian Symposium on Databases **SBBD '05, 2005**.
- [Carreira & Galhardas 2004] Carreira, P., & Galhardas, "*Efficient development of data migration transformations*", Demo paper, Proc. of the ACM Conference on the Management of Data, **SIGMOD'04**, **2004**.
- [Carreira et al. 2007b] Carreira, P., Galhardas, H., Pereira, J., Martins F. & Silva, Mário J., "*On the performance of One-to-Many Data Transformations*", Venkatesh Ganti, Felix Naumann (Eds.): Proceedings of the 5th International Workshop on Quality in Databases, **QDB 2007 at VLDB** Conference, **2007**.

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`

AUTHORS	TITLE	…
Periklis Andritsos, Ronald Fagin, Ariel Fuxman, Laura M. Haas, Mauricio A. Hernández, C. T. Howard Ho, Anastasios Kementsietsidis, Renée J. Miller, Felix Naumann, Lucian Popa, Yannis Velegrakis, Charlotte Vilarem, Ling-Ling Yan	Schema Management	…
Jorge Vieira, Jorge Bernardino, Henrique Madeira	Efficient Compression of Text Attributes of Data Warehouse Dimensions	…
…	…	…
…	…	…

Target: `EVENTS`

NAME	TITLE	…
Periklis Andritsos	Schema Management	…
Ronald Fagin	Schema Management	…
Ariel Fuxman	Schema Management	…
Laura M. Haas	Schema Management	…
Mauricio A. Hernández	Schema Management	…
C. T. Howard Ho	Schema Management	…
Anastasios Kementsietsidis	Schema Management	…
Renée J. Miller	Schema Management	…
Felix Naumann	Schema Management	…
Lucian Popa	Schema Management	…
Yannis Velegrakis	Schema Management	…
Charlotte Vilarem	Schema Management	…
Ling-Ling Yan	Schema Management	…
…	…	…

Example: Converting lines into columns

Source: `LOANEVT`

LOANO	EVTYP	CAPTL	TAX	EXPNS	BONUS
1234	OPEN	0.0	0.19	0.28	0.1
1234	PAY	1000.0	0.28	0.0	0.0
1234	PAY	1250.0	0.30	0.0	0.0
1234	EARLY	550.0	0.0	0.0	0.0
1234	FULL	5000.0	1.1	5.0	3.0
1234	CLOSED	0.0	0.1	0.0	0.0

Target: `EVENTS`

LOANO	EVTYP	AMTYP	AMT
1234	OPEN	TAX	0.19
1234	OPEN	EXPNS	0.28
1234	OPEN	BONUS	0.1
1234	PAY	CAPTL	1000.0
1234	PAY	TAX	0.28
1234	PAY	CAPTL	1250.0
1234	PAY	TAX	0.30
1234	EARLY	CAPTL	550.0
1234	FULL	CAPTL	5000.0
1234	FULL	TAX	1.1
1234	FULL	EXPNS	5.0
1234	FULL	BONUS	3.0
1234	CLOSED	EXPNS	0.1

Implementations of one-to-many data transformations

	Bounded				Unbounded			
	RA/SQL	PSM	Recursive Queries	Unpivot	RA/SQL	PSM	Recursive Queries	Unpivot
DBX	YES	YES	YES	N/A	NO	YES	YES	NO
OEX	YES	YES	N/A	N/A	NO	YES	N/A	NO

Extensions to Relational Algebra

Type of data transformation	Unary Extension to RA	Type of function
Many-to-one	Aggregation	Aggregate Function Set ⟶ Value
One-to-one	Extended Projection	Scalar Function Value ⟶ Value
One-to-many	**Mapper**	**Mapper Function** **Value ⟶ Set**

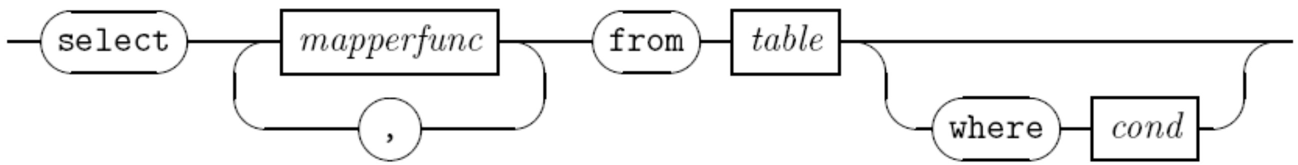
Mapper example

Extra slide 4

```
select
  lpad(tostr(ACCT), 4, '0') as ACCTNO,
  map AMOUNT, SEQNO
    begin
      var rem_amnt: numeric
      var seq_no: integer  = 0
      rem_amnt = AMT
      loop while rem_amnt > 100 do
        rem_amnt = rem_amnt - 100
        seq_no = seq_no + 1
        insert rem_amnt, seq_no
      end loop
      if rem_amnt > 0 then
        insert rem_amnt, seq_no + 1
      end if
    end
from LOANS
```

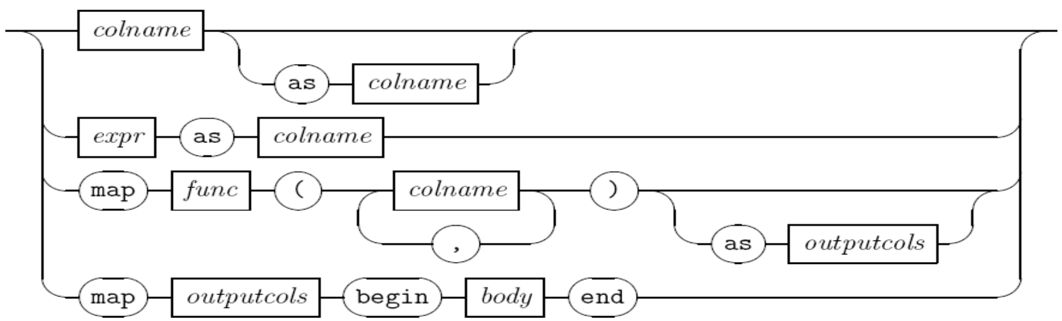
Simplified SELECT syntax for mappers

select

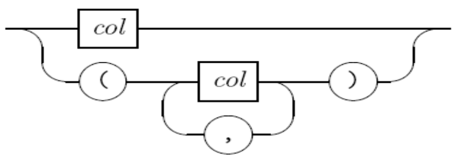


Syntactic details for mapper functions

mapperfunc

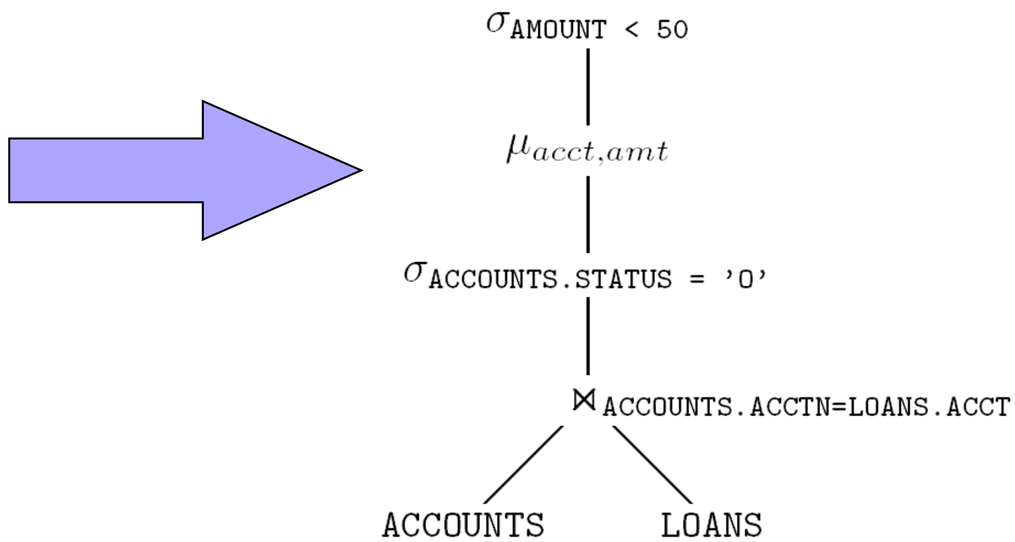


outputcols



Example: Query with the mapper operator

```
select map acct(ACCT) as ACCTNO,
       map amt(AM) as AMOUNT
from LOANS, ACCOUNTS
where ACCOUNTS.ACCTN = LOANS.ACCT
  and ACCOUNTS.STATUS = 'O'
  and AMOUNT < 50
```



Union

```
insert into EVENTS (LOANNO, EVTYP, AMTYP, AMT)
  select LOANNO, EVTYP, 'CAPTL' as AMTYP, CAPTL
     from LOANEVT
     where CAPTL > 0
   union all
   select LOANNO, EVTYP, 'TAX' as AMTYP, TAX
     from LOANEVT
     where TAX > 0
   union all
   select LOANNO, EVTYP, 'EXPNS' as AMTYP, EXPNS
     from LOANEVT
     where EXPNS > 0
   union all
   select LOANNO, EVTYP, 'BONUS' as AMTYP, BONUS
     from LOANEVT
    where BONUS > 0;
```

Recursive Query

```
with recpayments(digits(ACCTNO), AMOUNT, SEQNO, REMAMNT) as
   (select ACCT,
          case when base.AM < 100 then base.AM
            else 100 end,
          1,
          case when base.AM < 100 then 0
            else base.AM - 100 end
    from LOANS as base
  union all
    select ACCTNO,
          case when step.REMAMNT < 100 then
              step.REMAMNT
            else 100 end,
          SEQNO + 1,
          case when step.REMAMNT < 100 then 0
            else step.REMAMNT - 100 end,
      from recpayments as step
      where step.REMAMNT > 0)
  select ACCTNO, SEQNO, AMOUNT
  from recpayments as PAYMENTS
```

Table function

```
create function LOANSTOPAYMENTS
  return PAYMENTS_TABLE_TYPE pipelined is
  ACCTVALUE LOANS.ACCT%TYPE;
  AMVALUE LOANS.AM%TYPE;
  REMAMNT INT;
  SEQNUM INT;
  cursor CLOANS is
    select * from LOANS;
begin
    open CLOANS;
    loop
      fetch CLOANS into ACCTVALUE, AMVALUE;
      REMAMNT := AMVALUE;
      SEQNUM := 1;
      while REMAMNT > 100
        loop
          pipe row(PAYMENTS_ROW_TYPE(
            LPAD(ACCTVALUE, 4, '0'), 100.00, SEQNUM));
          REMAMNT := REMAMNT - 100;
          SEQNUM := SEQNUM + 1;
        end loop
      if REMAMNT > 0 then
        pipe row(PAYMENTS_ROW_TYPE(
          values (LPAD(ACCTVALUE, 4, '0'),
            REMAMNT, SEQNUM));
      end if
  end loop
end LOANSTOPAYMENTS
```

An iterator that scans the input relation once

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

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

$$\sigma_{C_{A_i}}(\mu_F(s)) = \mu_{F \setminus \{f_{A_i}\} \cup \{\sigma_{C_{A_i}} \circ f_{A_i}\}}(s)$$

Distributing mappers over unions

$$\mu_F(r \cup s) = \mu_F(r) \cup \mu_F(s)$$

Distributing mappers over Cartesian products

$$\mu_F(s \times r) = \mu_{F_S}(s) \times \mu_{F_R}(r)$$

Statistics for computing the cost of One-to-many data transformations

- Given a mapper μ_F
  1. Average selectivity of the mapper (α_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

$$card(\mu_F(e)) = card(e) \cdot \alpha_F \cdot O_F$$

O_F: Fanout factor

α_F: Selectivity factor

$card(e)$: number of input tuples

$card(e) \cdot \alpha_F$: input tuples that are not filtered

$card(e) \cdot \alpha_F \cdot O_F$: total number of output tuples

- Estimating the CPU cost of a mapper expression

$$cost_{\text{CPU}}(\mu_F(e)) = card(e) \cdot (k \cdot O_F + C_F)$$

$card(e)$: number of input tuples

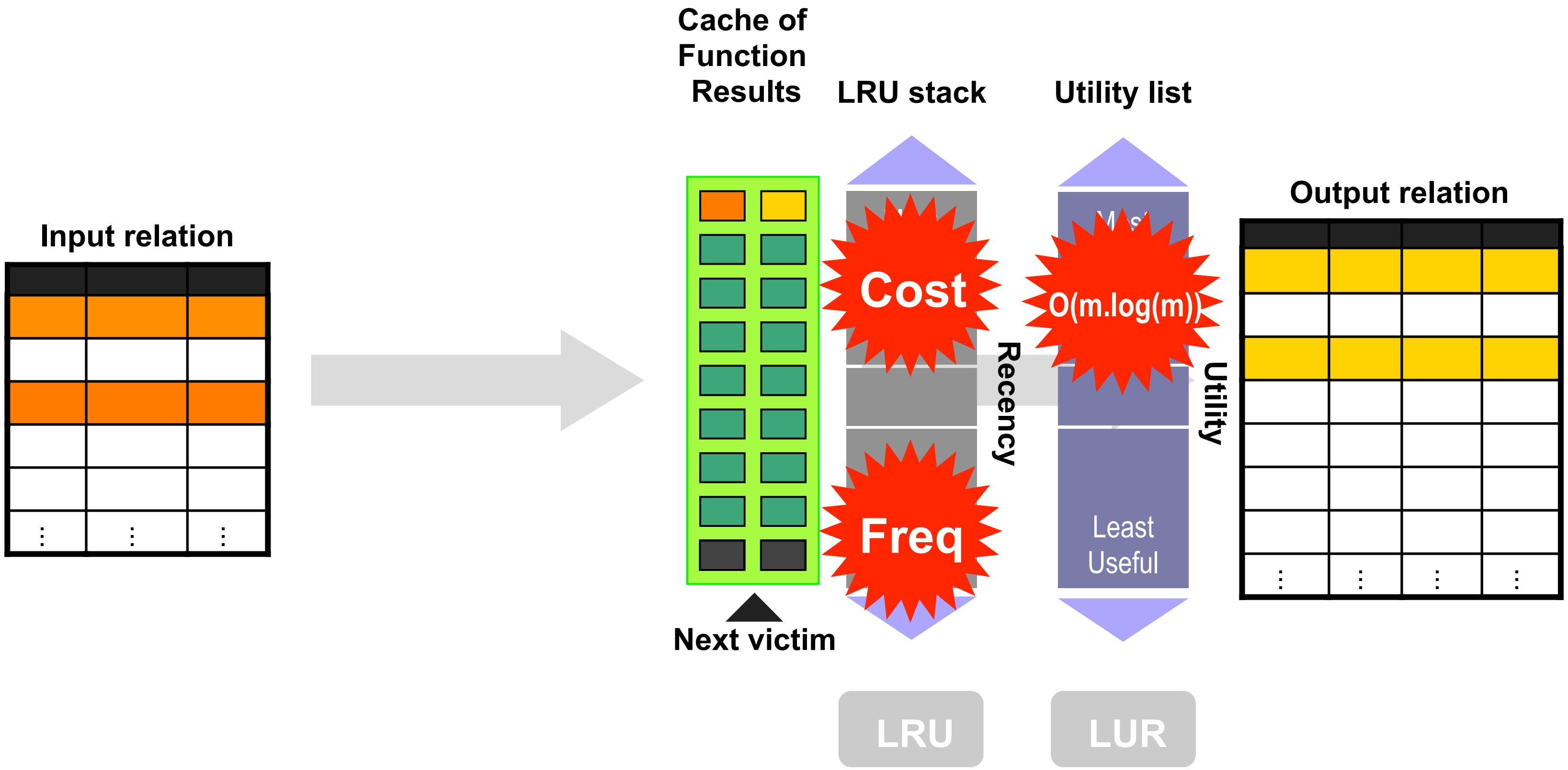
$k \cdot O_F$: CPU cost of the cartesian product operation

C_F: CPU cost of evaluating all the mapper functions

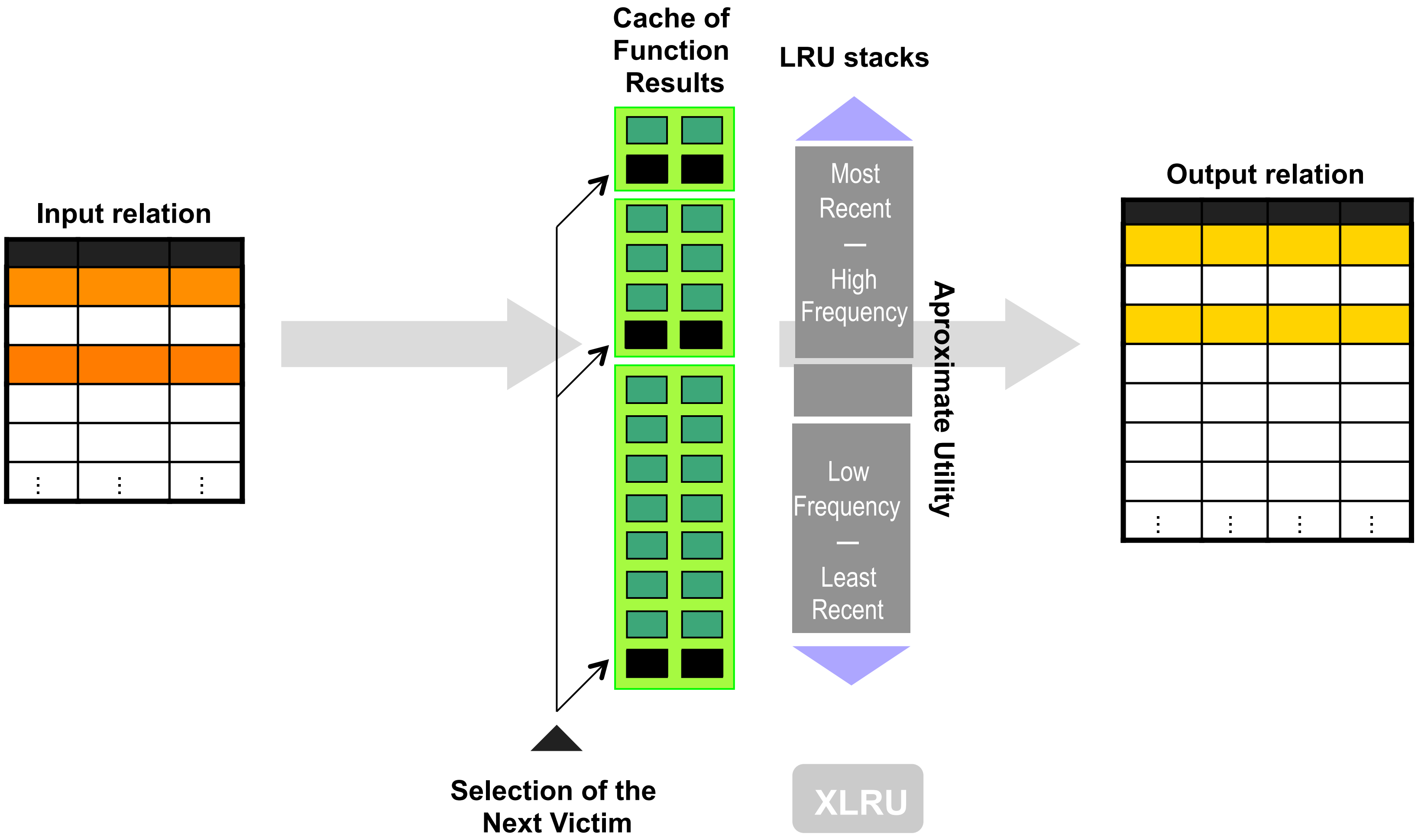
$(k \cdot O_F + C_F)$: per-tuple CPU cost

$card(e) \cdot (k \cdot O_F + C_F)$: total CPU cost

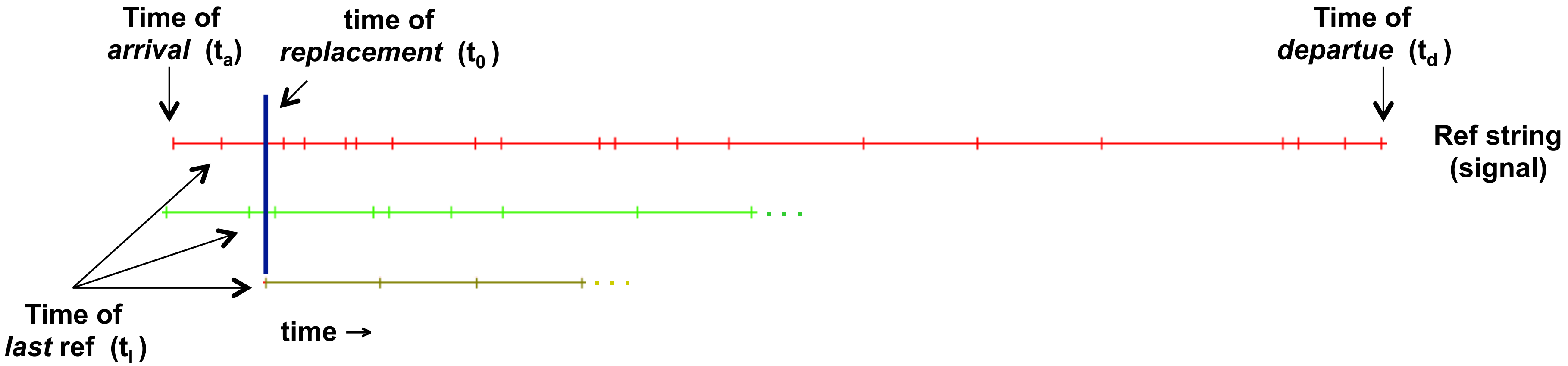
Cache-based evaluation 1/2



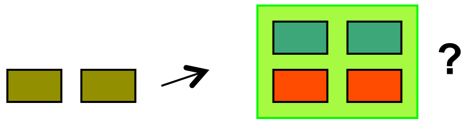
Cache-based evaluation 2/2



An utility metric based on statistical inference

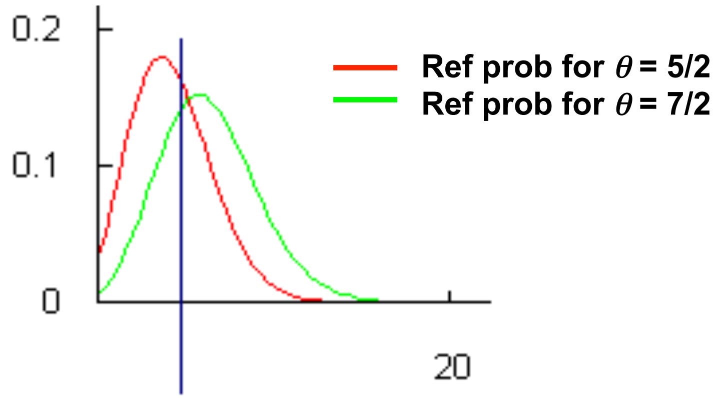


- Which entry to replace?



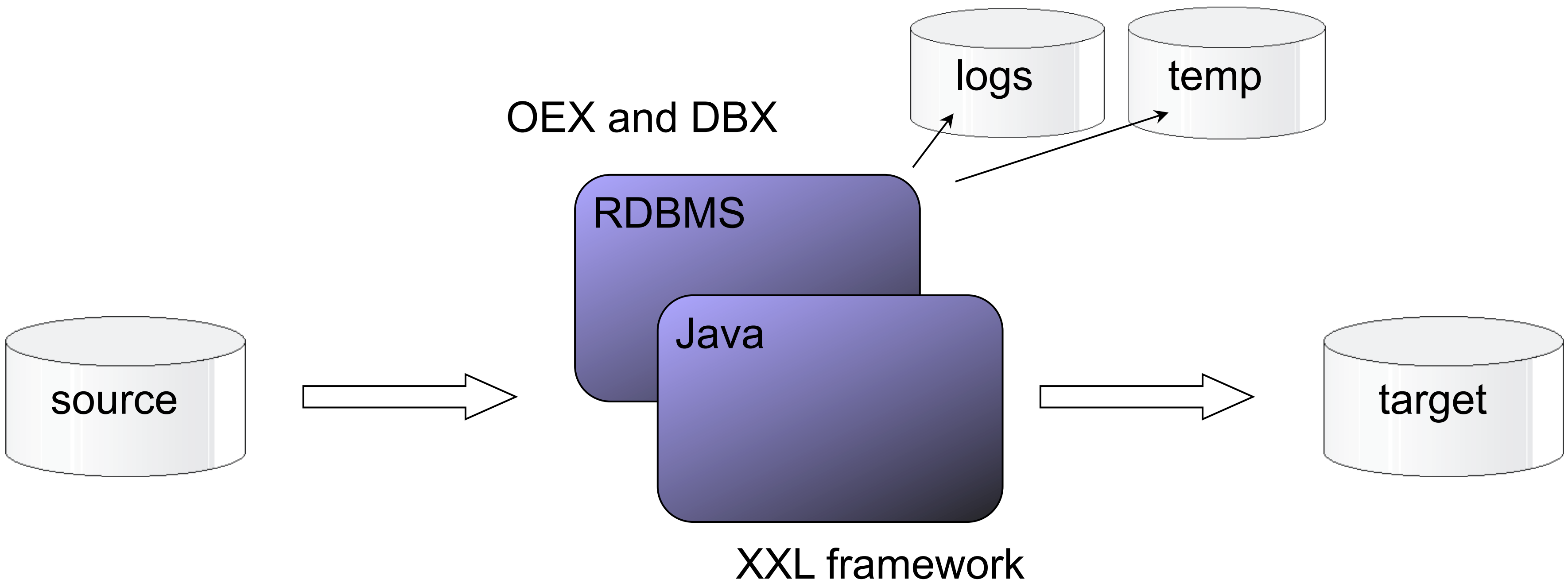
- Utility $u_{t0}(e)$ based on:
  - Number of past references n_h
  - Cost c
  - Avg frequency θ
  - Time to last $k = (t_0 - t_l)$

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



Extra slide 2

Experimental Setup



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

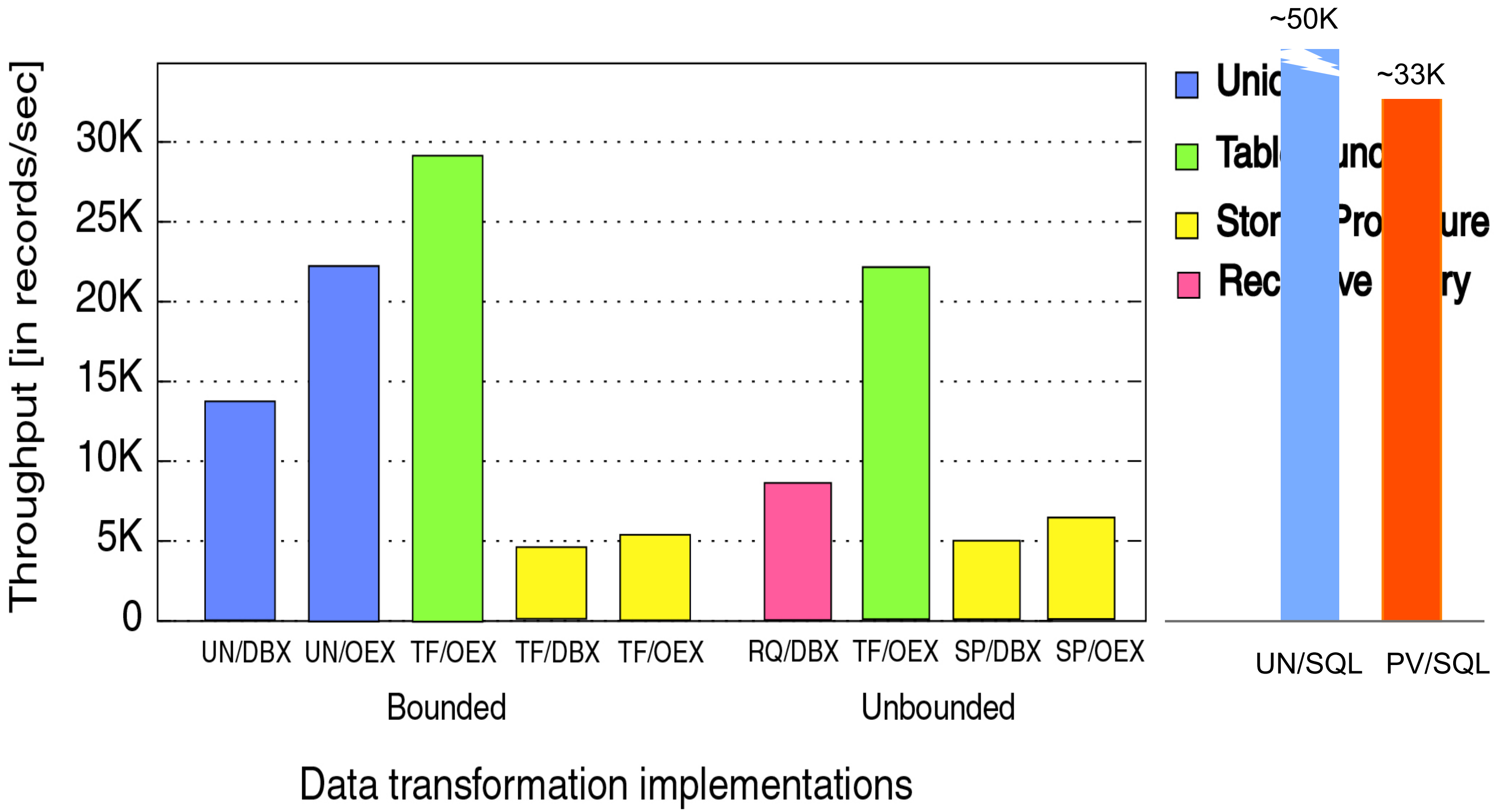
Settings checklist

- 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

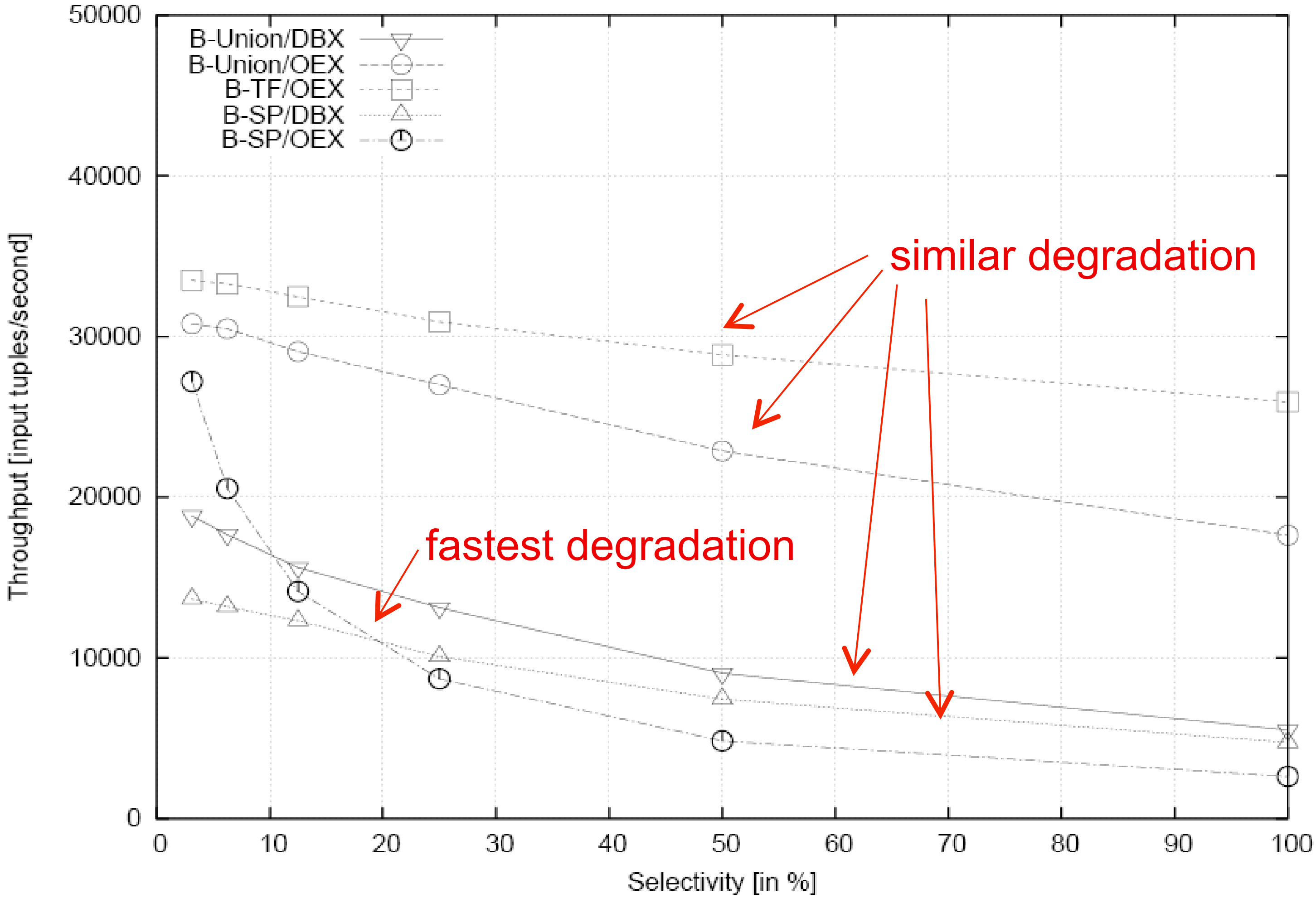
Resource consumption

	Union	Unpivot	RQ	TF
> fanout	more input more output bigger query	input once more output	input once more output more *'temp'*	input once more output
> select	same input more output	same input more output	same input more *'temp'* more output	same input more output

Throughput of one-to-many transformations



Influence of selectivity



Original *v.s.* optimized expression