Incremental Data Transformations on Wide-Column Stores with NoTaQL

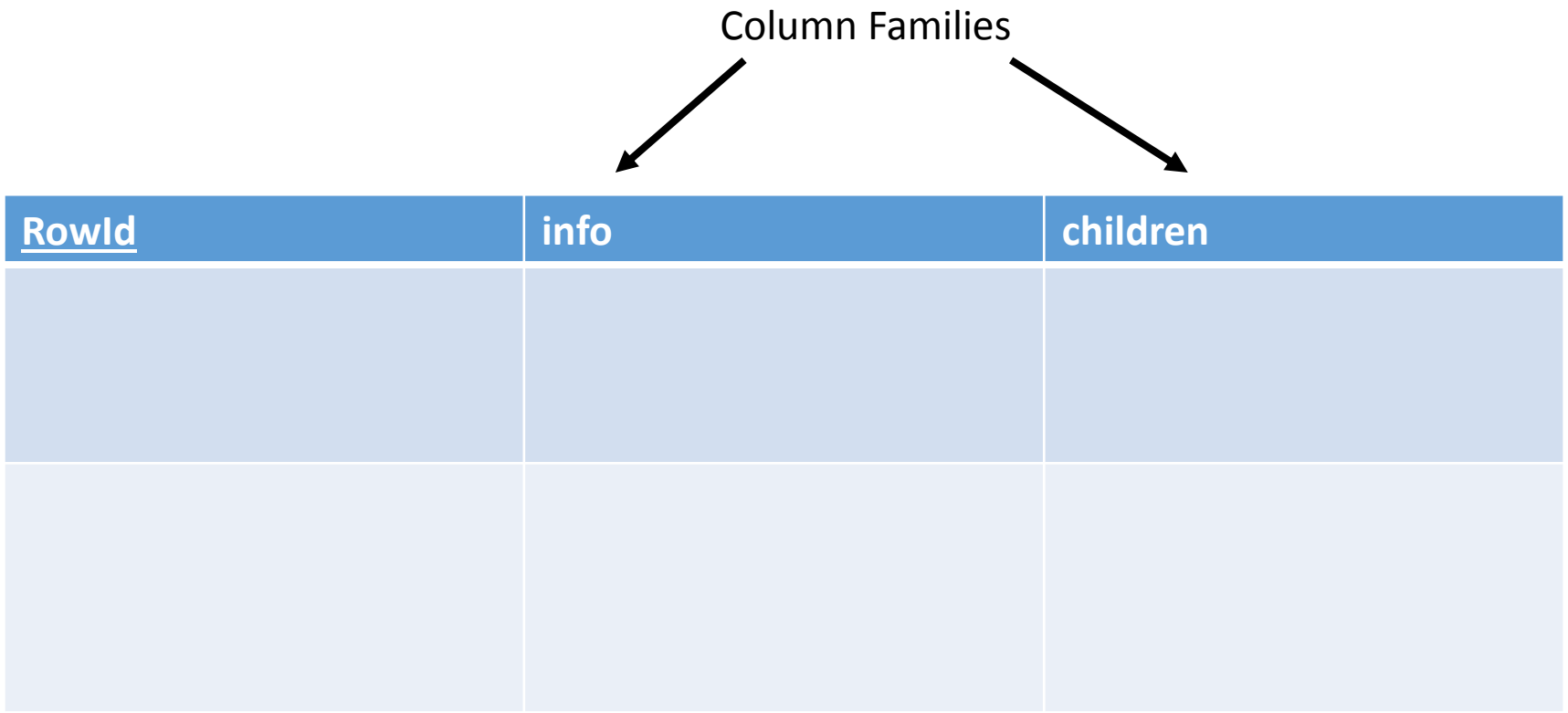
2014-06-29

M. Sc. Johannes Schildgen

schildgen@cs.uni-kl.de

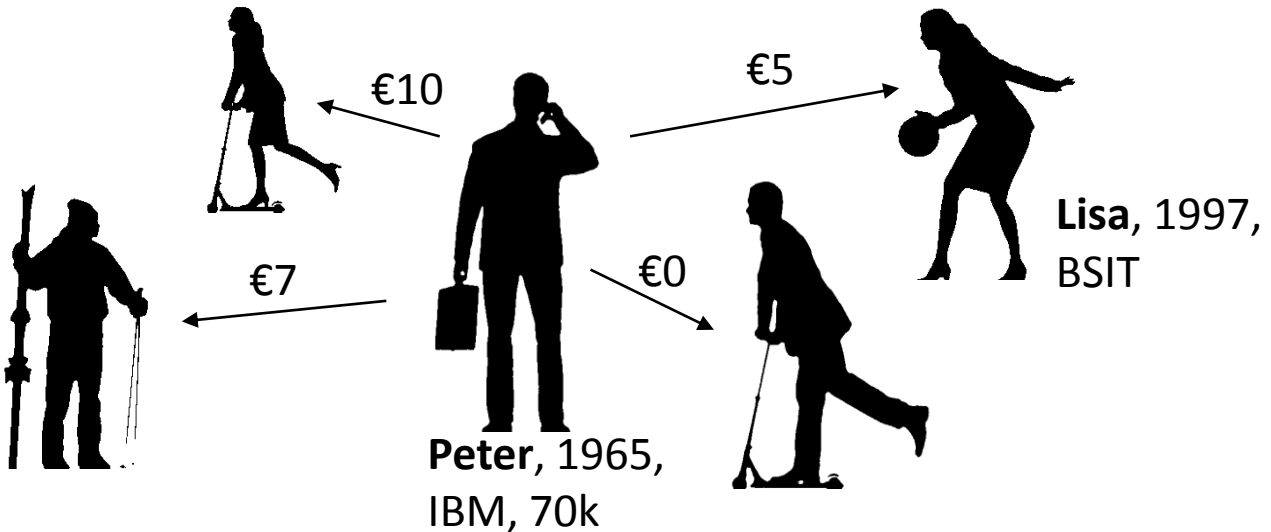
TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

"A DBA walks into a NoSQL bar, but turns and leaves because he couldn't find a table"


Column Families
RowId
info
children

Column Families

RowId	info			children			
Peter	born	cmpny	salary	Lisa	Carl	Susi	Toni
	1965	IBM	70k	€5	€0	€10	€7
Lisa	born	school					
	1997	BSIT					



HBase API

```
put ‘pers‘, ‘Carl‘,
    ‘info:born‘, ‘1982‘

put ‘pers‘, ‘Carl‘,
    ‘info:school‘, ‘BSIT‘

put ‘pers‘, ‘Carl‘,
    ‘info:school‘, ‘BUIT‘

get ‘pers‘, ‘Carl‘
```

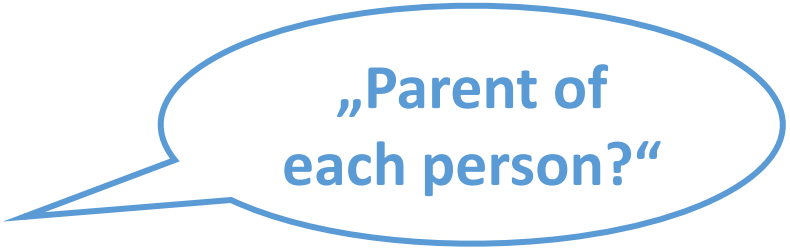
Jaspersoft HBase QL


```
{
    "tableName": "pers",
    "deserializerClass": "com.jaspersoft…DefaultDeserializer",
    "filter": {
        "SingleColumnValueFilter": {
            "family": „info",
            "qualifier": „school",
            "compareOp": "EQUAL",
            "comparator": { "SubstringComparator": { "substr":
                           „BSIT" } }
        }
    }
}
```

http://community.jaspersoft.com/wiki/jaspersoft-hbase-query-language

Phoenix

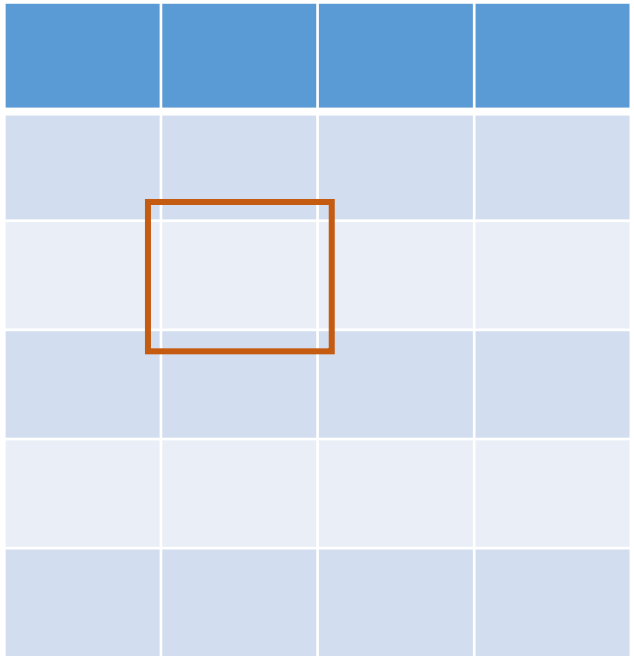

```
SELECT * FROM pers WHERE school = 'BSIT'
```



https://github.com/forcedotcom/phoenix

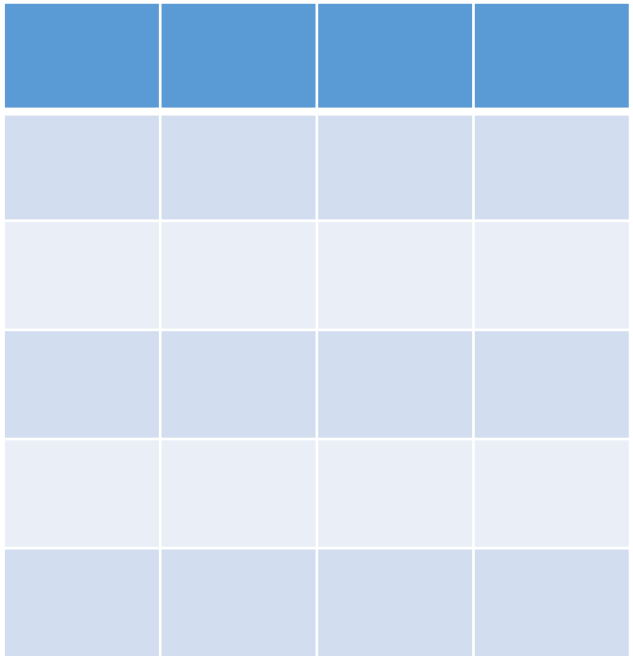

NôtaQL

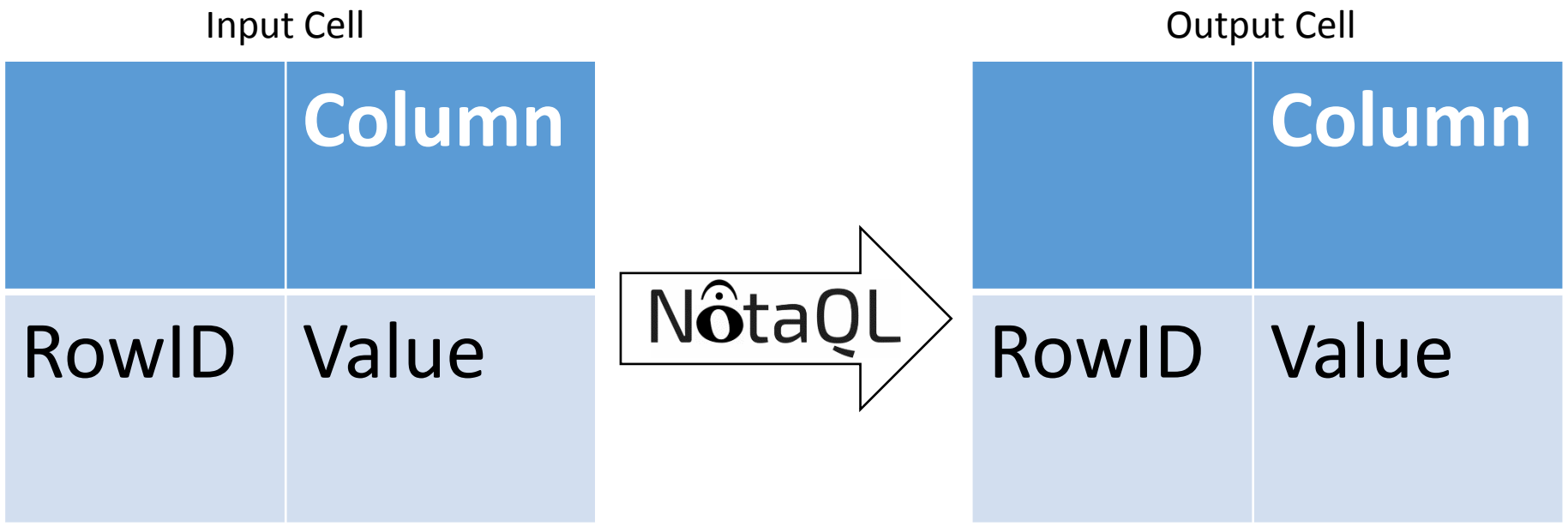
Input Table

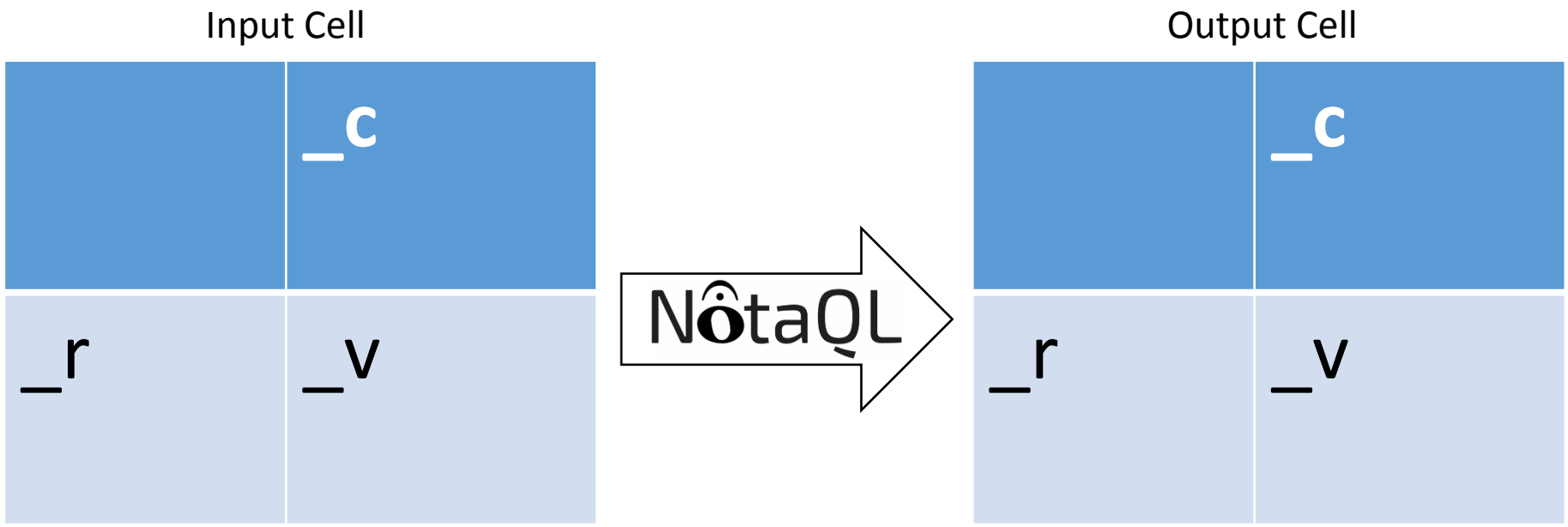


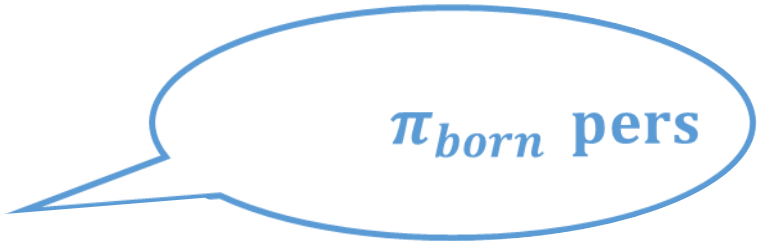
NotaQL

Output Table




Input Cell
Column
RowID
Value
NotaQL
Output Cell
Column
RowID
Value


Input Cell
_c
_r
_v
NotaQL
Output Cell
_c
_r
_v



Input Cell

	_c *born*
_r *Lisa*	_v *1997*

Output Cell

	_c *born*
_r *Lisa*	_v *1997*

NotaQL

```
OUT._r <- IN._r,
OUT.born <- IN.born;
```

Input Cell

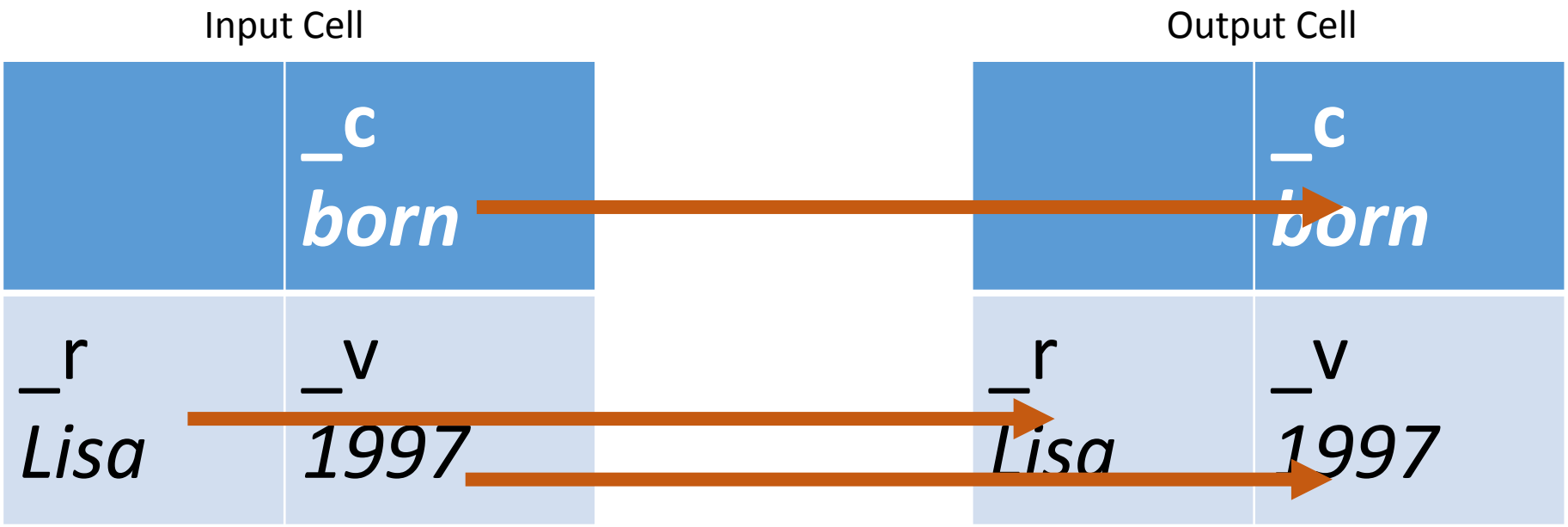
	_c *born*
_r *Lisa*	_v *1997*

Output Cell

	_c *born*
_r *Lisa*	_v *1997*

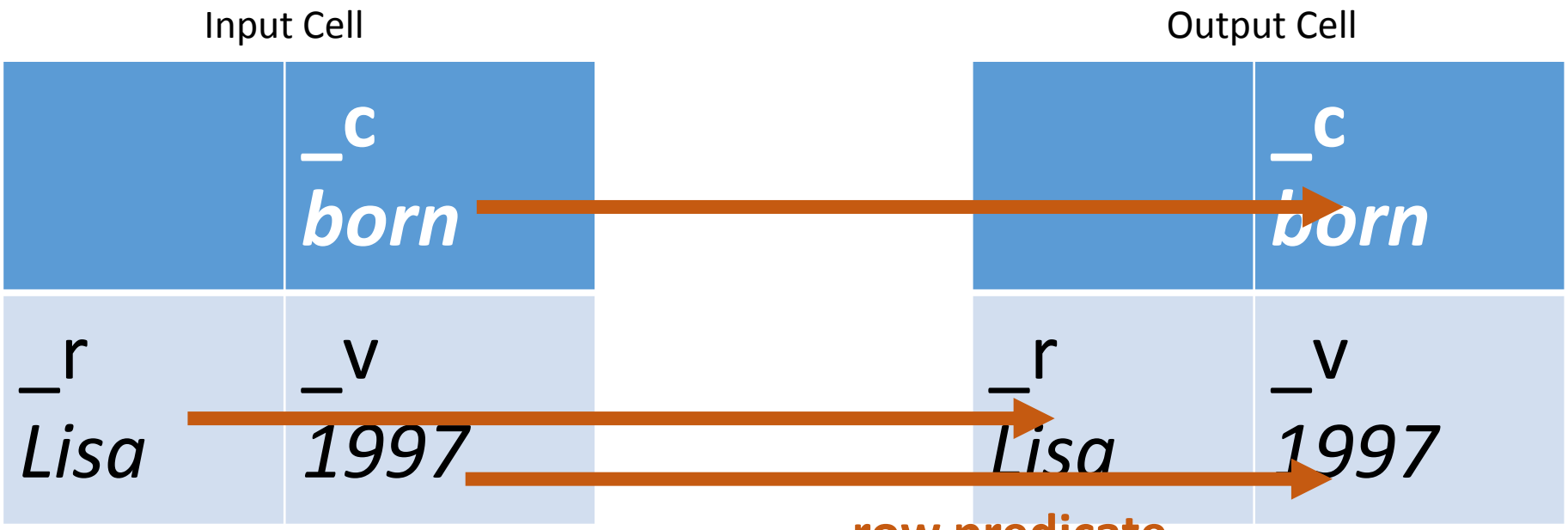
NotaQL

```
OUT._r <- IN._r,
OUT.born <- IN.born,
OUT.school <- IN.school;
```



NotaQL

```
OUT._r <- IN._r,
OUT.$(IN._c) <- IN._v;
```



row predicate

NotaQL

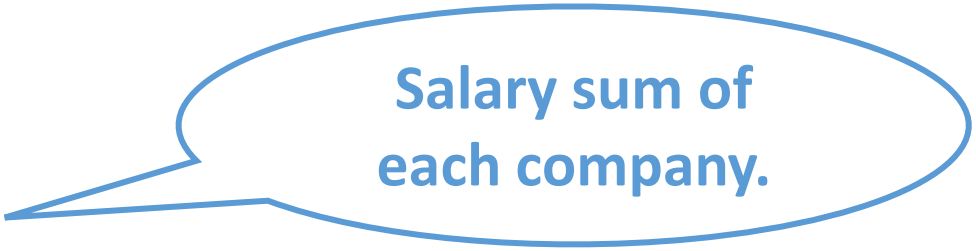
```
IN-FILTER: school='BSIT',
OUT._r <- IN._r,
OUT.$(IN._c) <- IN._v;
```

That was:

Selection and Projection

Now:

Grouping



Input Cell

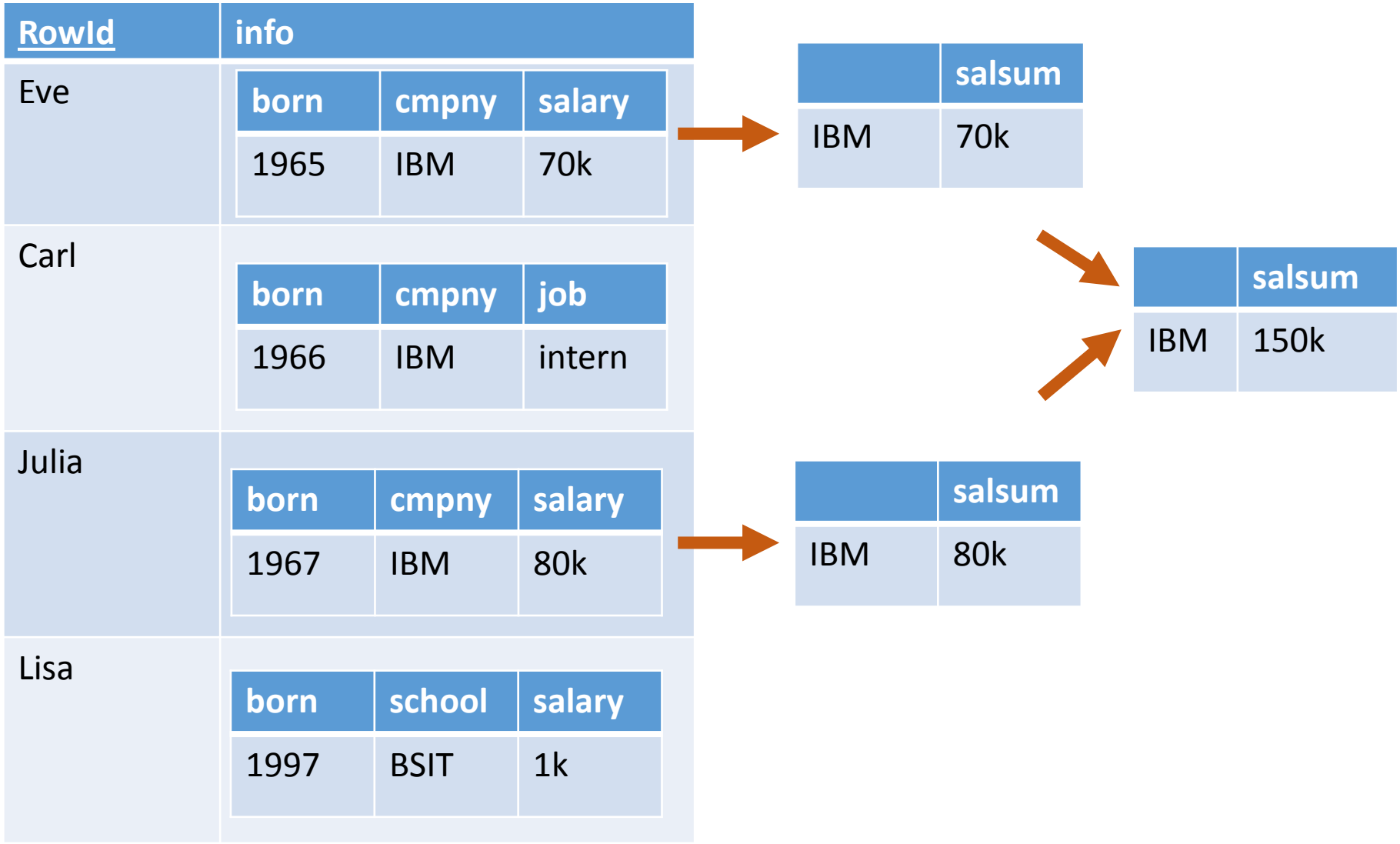
	_c *cmpny*
_r *Peter*	_v *IBM*

Output Cell

	_c *salsum*
_r *IBM*	_v *645k*

NoTaQL

```
OUT._r <- IN.cmpny,
OUT.salsum <- SUM(IN.salary);
```

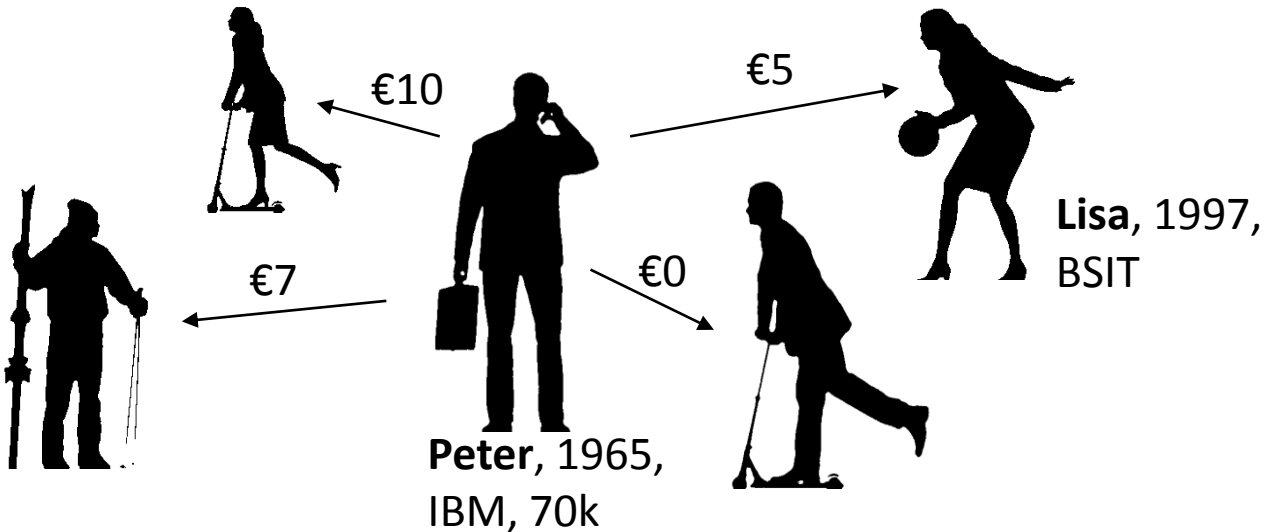


NotaQL

```
OUT._r <- IN.cmpny,
OUT.salsum <- SUM(IN.salary):
```

Advanced Transformations:
More Filters

RowId	info			children			
Peter	born	cmpny	salary	Lisa	Carl	Susi	Toni
	1965	IBM	70k	€5	€0	€10	€7
Lisa	born	school					
	1997	BSIT					



RowId	info			children			
Peter	born	cmpny	salary	Lisa	Carl	Susi	Toni
	1965	IBM	70k	€5	€0	€10	€7
Lisa	born	school					
	1997	BSIT					

NotaQL

```
OUT._r <- IN._r,
OUT.$(IN._c) <- IN._v;
```

RowId	info			children			
Peter	born	cmpny	salary	Lisa	Carl	Susi	Toni
	1965	IBM	70k	€5	€0	€10	€7
Lisa	born	school					
	1997	BSIT					

NotaQL

```
IN-FILTER: COL_COUNT(children)>0
OUT._r <- IN._r,
OUT.$(IN._c) <- IN._v;
```

RowId	info	children
Peter	born: 1965; cmpny: IBM; salary: 70k	Lisa: €5; Carl: €0; Susi: €10; Toni: €7
Lisa	born: 1997; school: BSIT	

NotaQL

```
IN-FILTER: COL_COUNT(children)>0
OUT._r <- IN._r,
OUT.$(IN.children._c) <- IN._v;
```

RowId	info	children
Peter	born: 1965; cmpny: IBM; salary: 70k	Lisa: €5; Carl: €0; Susi: €10; Toni: €7
Lisa	born: 1997; school: BSIT	

NotaQL

```
IN-FILTER: COL_COUNT(children)>0
OUT._r <- IN._r,
OUT.$(IN.children._c?(@>5))
                    <- IN._v;
```

cell predicate

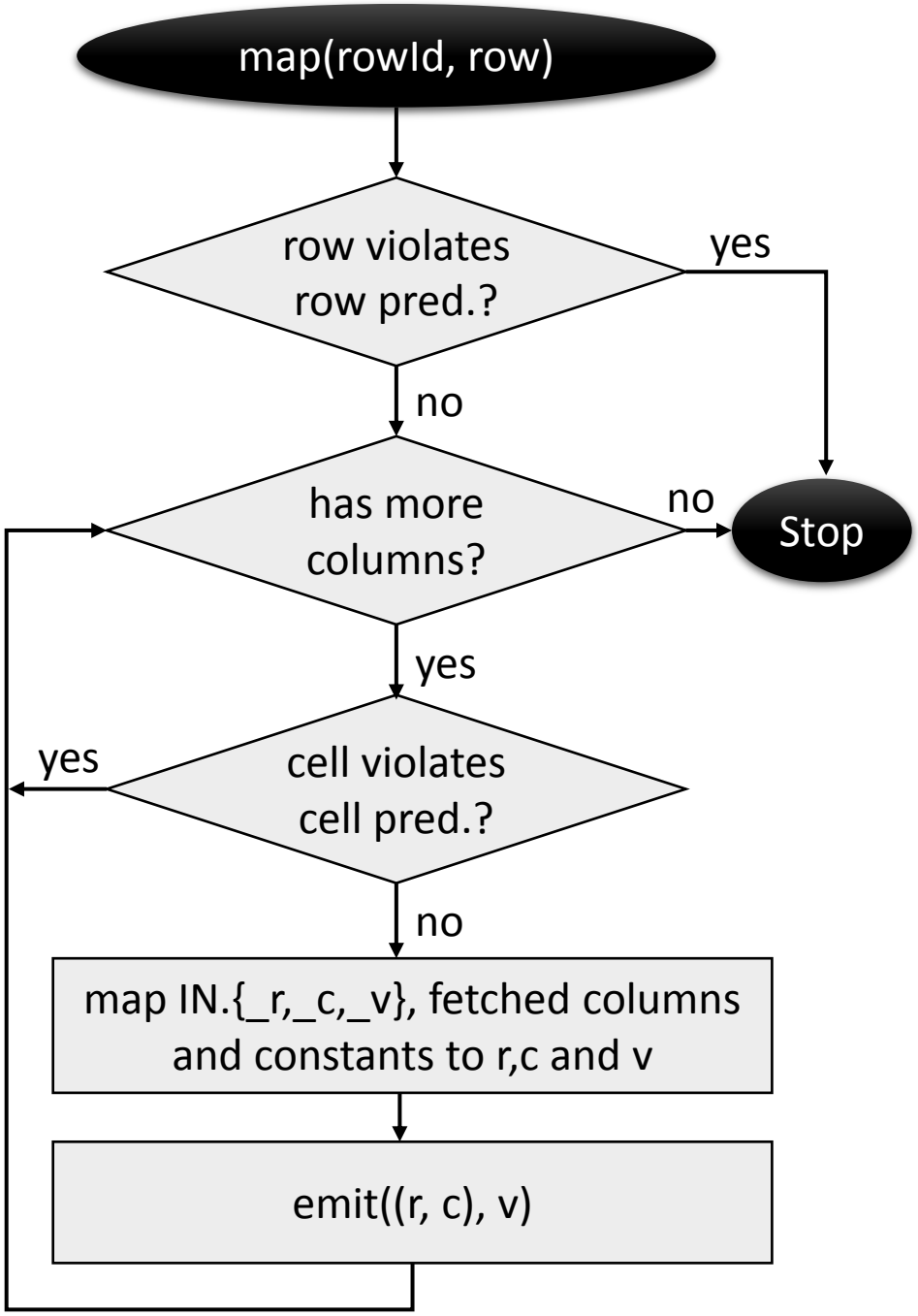
RowId	info	children
Peter	born: 1965; cmpny: IBM; salary: 70k	Lisa: €5; Carl: €0; Susi: €10; Toni: €7
Lisa	born: 1997; school: BSIT	

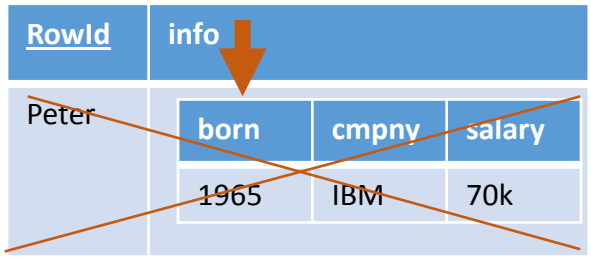
NotaQL

```
IN-FILTER: COL_COUNT(children)>0
OUT._r <- IN._r,
OUT.$(IN.children._c?(!Carl))
                    <- IN._v;
```

cell predicate

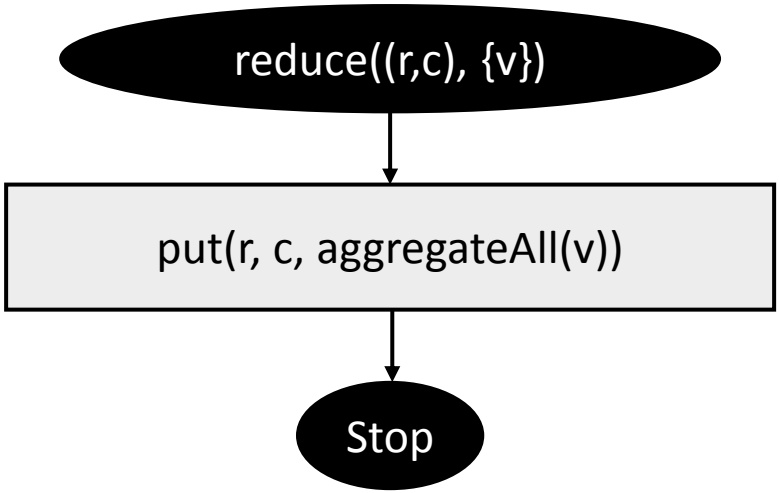
NotaQL Transformation Platform: **MapReduce**





	salsum
IBM	70k

((IBM, salsum), 70k)



((IBM, salsum), {70k, 80k, 10k})

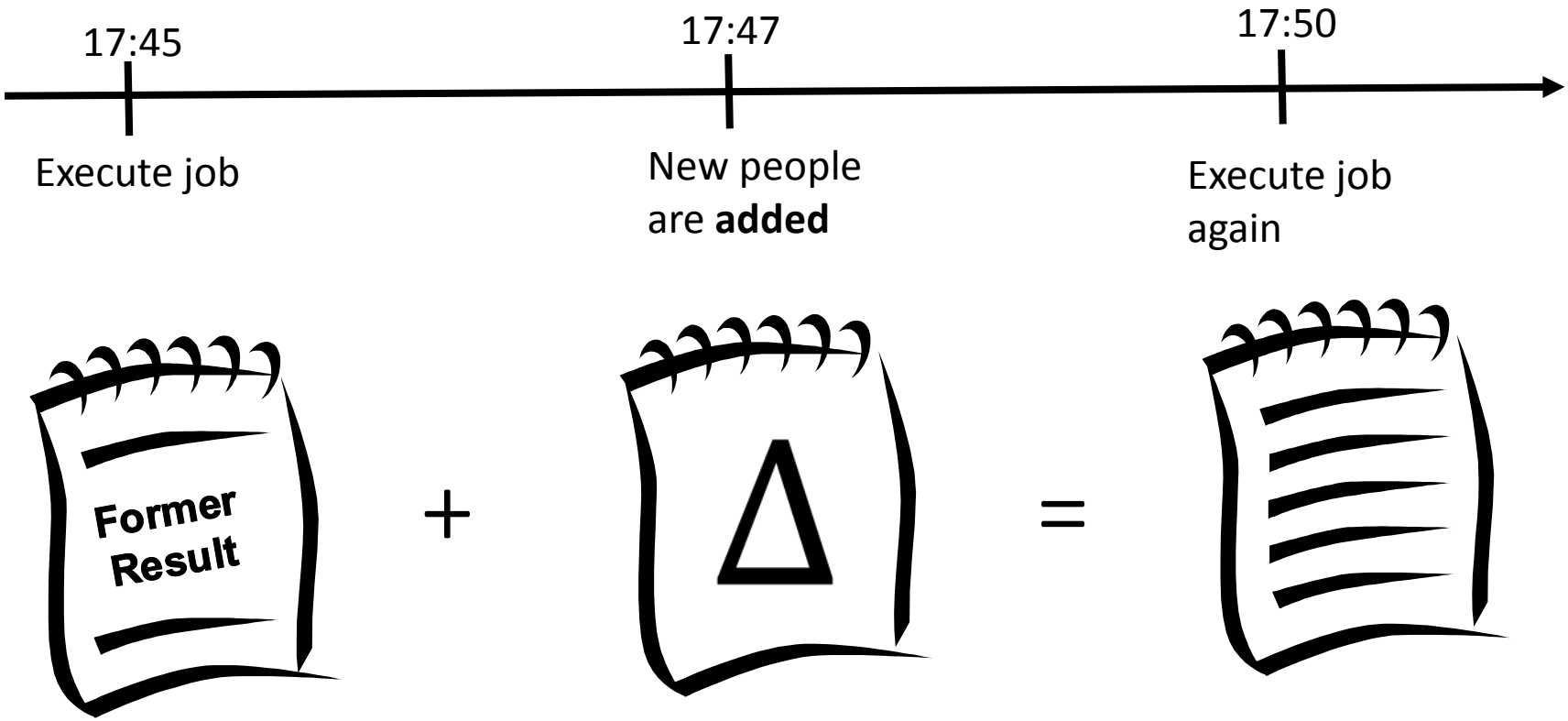
((IBM, salsum), 160k)

Incremental Transformations: **Self-Maintainability**

Salary sum of all people born before 1980 per company.

NotaQL

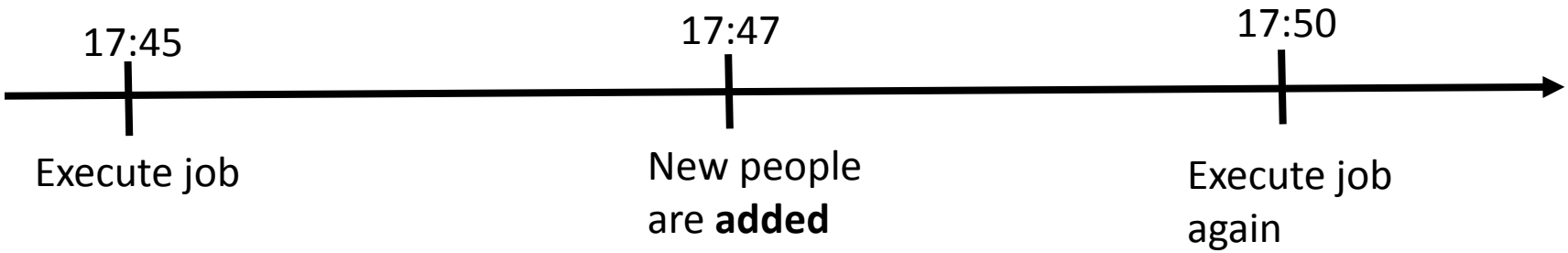
```
IN-FILTER: born<1980,
OUT._r <- IN.cmpny,
OUT.salsum <- SUM(IN.salary);
```



Salary sum of all people born before 1980 per company.

NotaQL

```
IN-FILTER: born<1980,
OUT._r <- IN.cmpny,
OUT.salsum <- SUM(IN.salary);
```



	salsum
IBM	150k

	born	cmpny	salary
~~Melissa~~	~~1989~~	~~IBM~~	~~50k~~
Nora	1977	IBM	80k

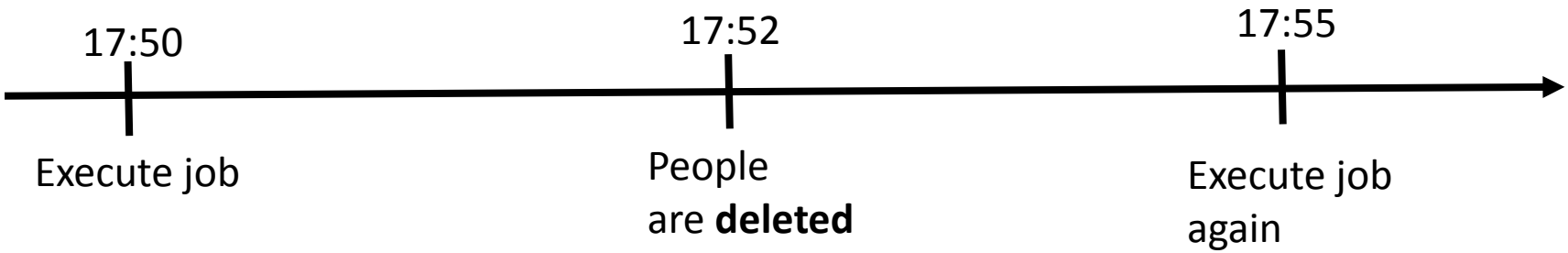
+

	salsum
IBM	230k

Salary sum of all people born before 1980 per company.

NotaQL

```
IN-FILTER: born<1980,
OUT._r <- IN.cmpny,
OUT.salsum <- SUM(IN.salary);
```



	salsum
IBM	230k

	born	cmpny	salary
~~Melissa~~	~~1989~~	~~IBM~~	~~50k~~
Nora	1977	IBM	80k

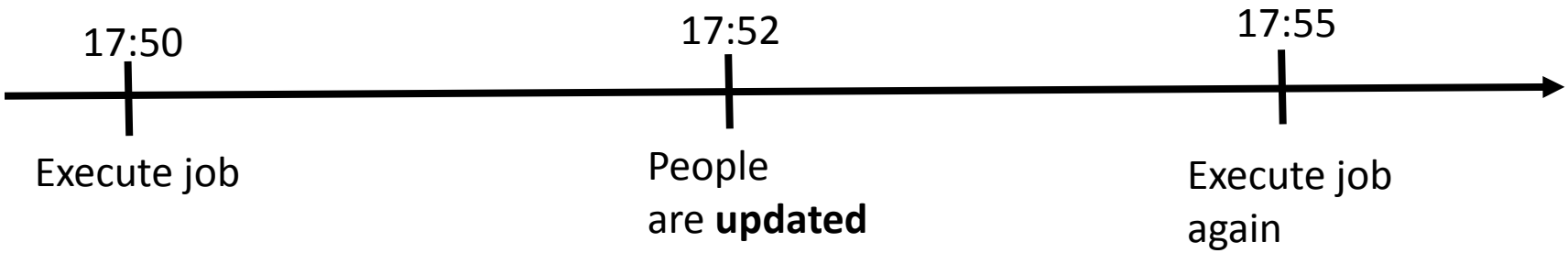
\-

	salsum
IBM	150k

Salary sum of all people born before 1980 per company.

NotaQL

```
IN-FILTER: born<1980,
OUT._r <- IN.cmpny,
OUT.salsum <- SUM(IN.salary);
```



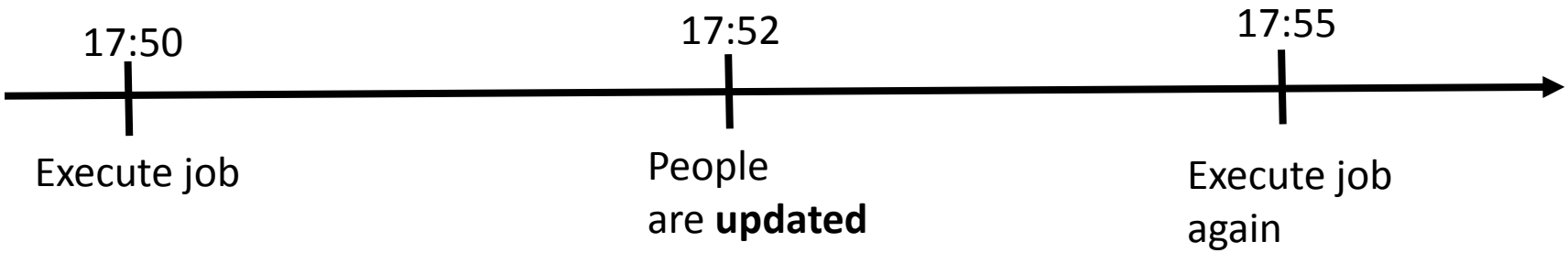
	born	cmpny	salary
Peter	~~1965~~ 1990	IBM	70k

-= 70k

Salary sum of all people born before 1980 per company.

NoitaQL

```
IN-FILTER: born<1980,
OUT._r <- IN.cmpny,
OUT.salsum <- SUM(IN.salary);
```



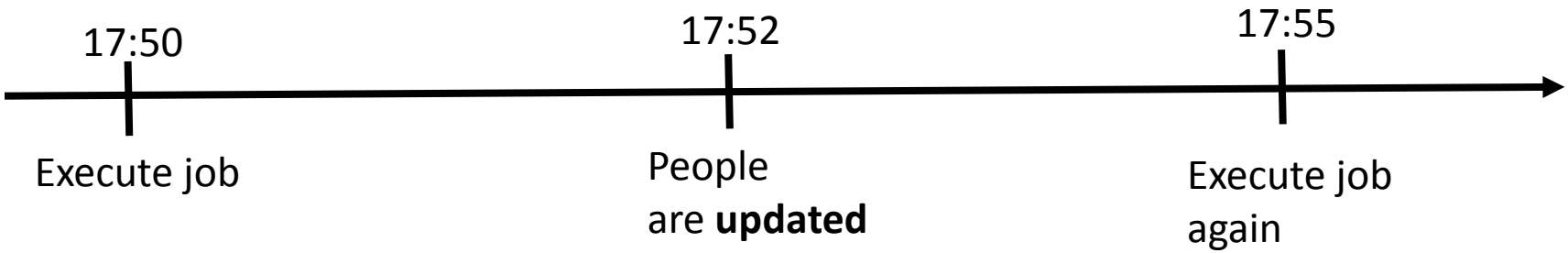
	born	cmpny	salary
Peter	~~1965~~ ~~1990~~ 1965	IBM	70k

+= 70k

Salary sum of all people born before 1980 per company.

NoitaQL

```
IN-FILTER: born<1980,
OUT._r <- IN.cmpny,
OUT.salsum <- SUM(IN.salary);
```



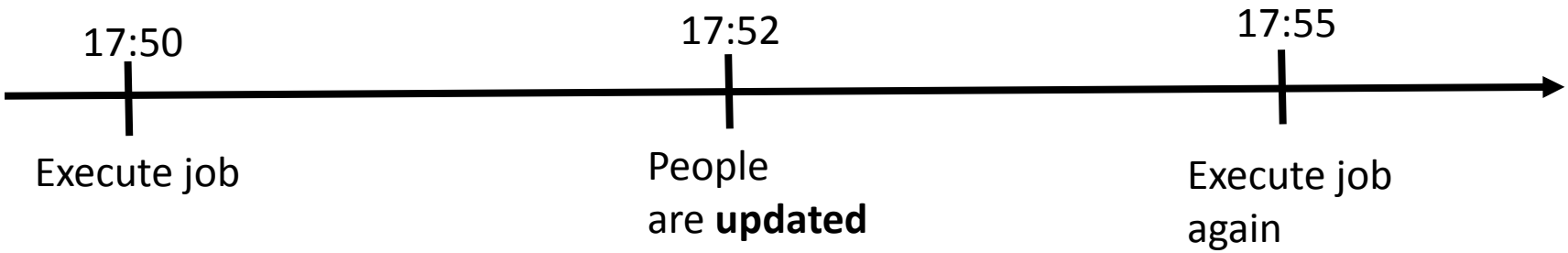
	born	cmpny	salary
Peter	1965	IBM	~~70k~~ 75k

+= 75k-70k

Salary sum of all people born before 1980 per company.

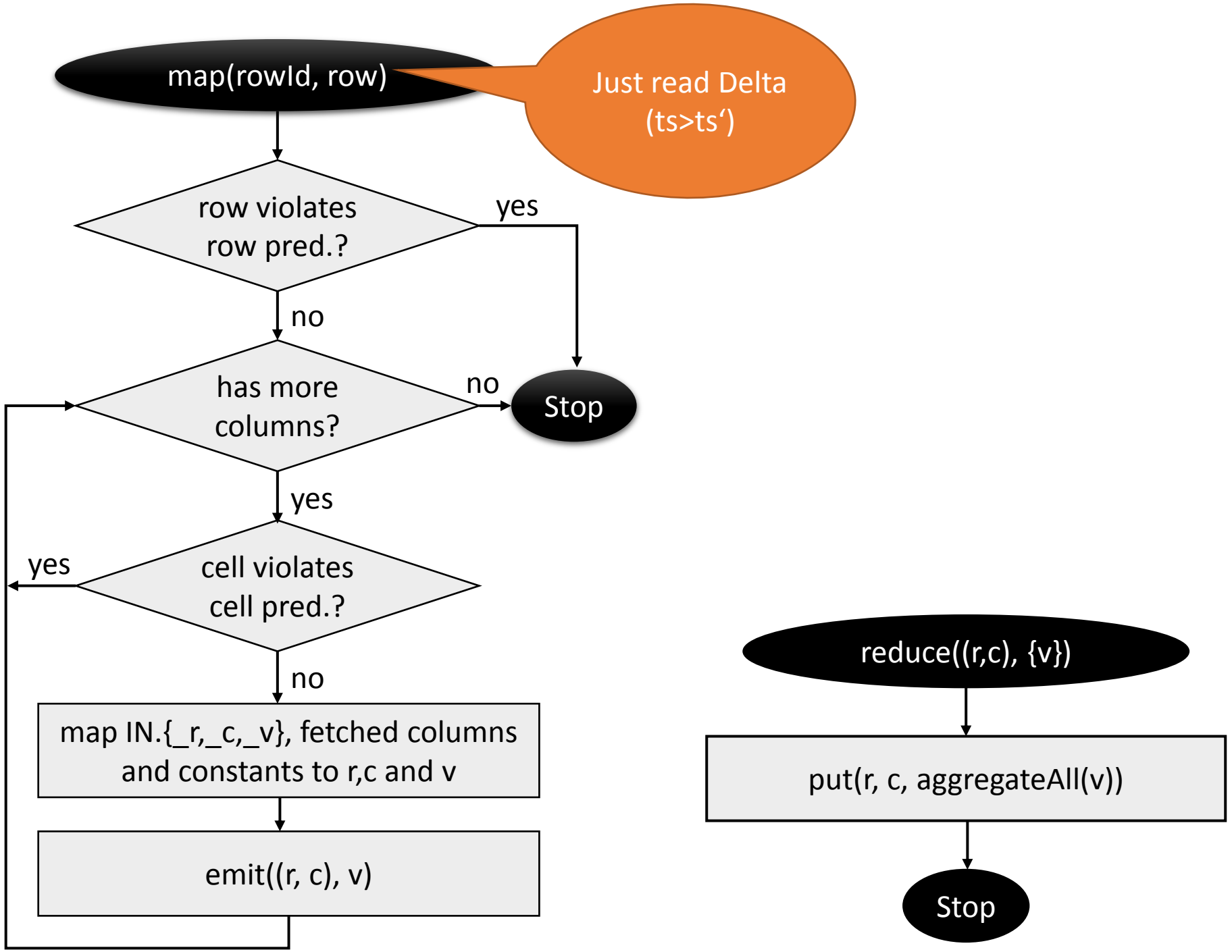
NotaQL

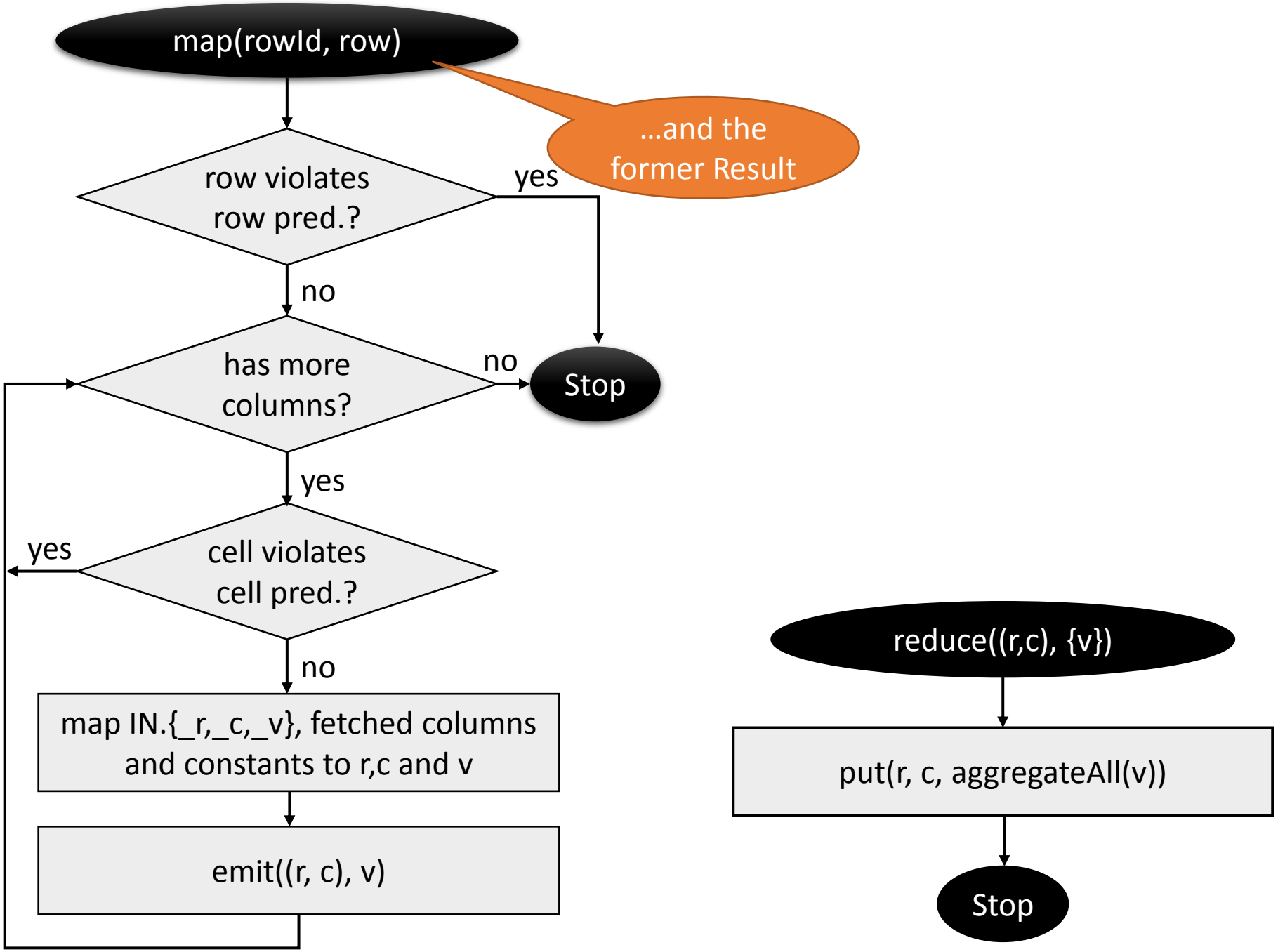
```
IN-FILTER: born<1980,
OUT._r <- IN.cmpny,
OUT.salsum <- SUM(IN.salary);
```

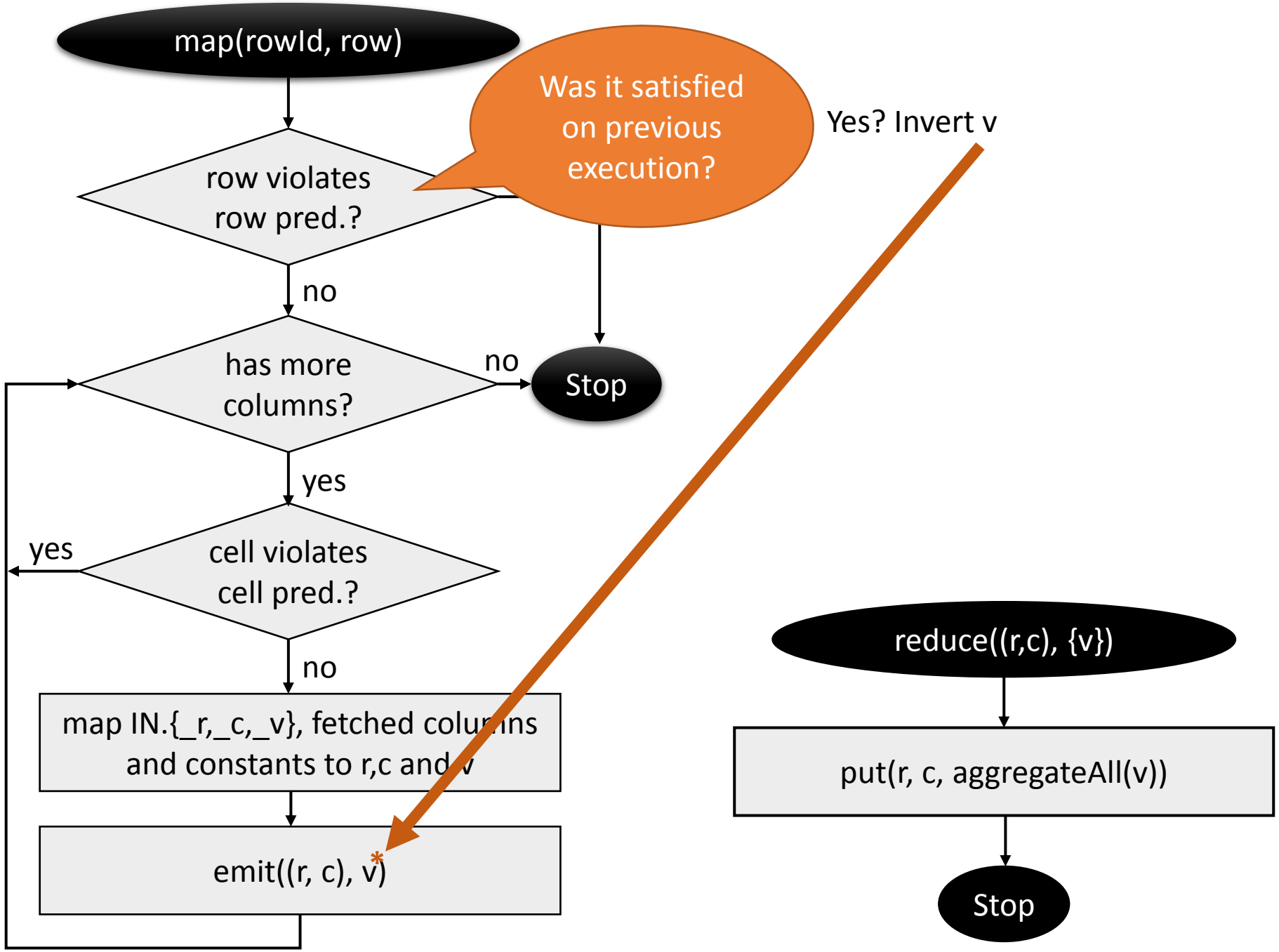


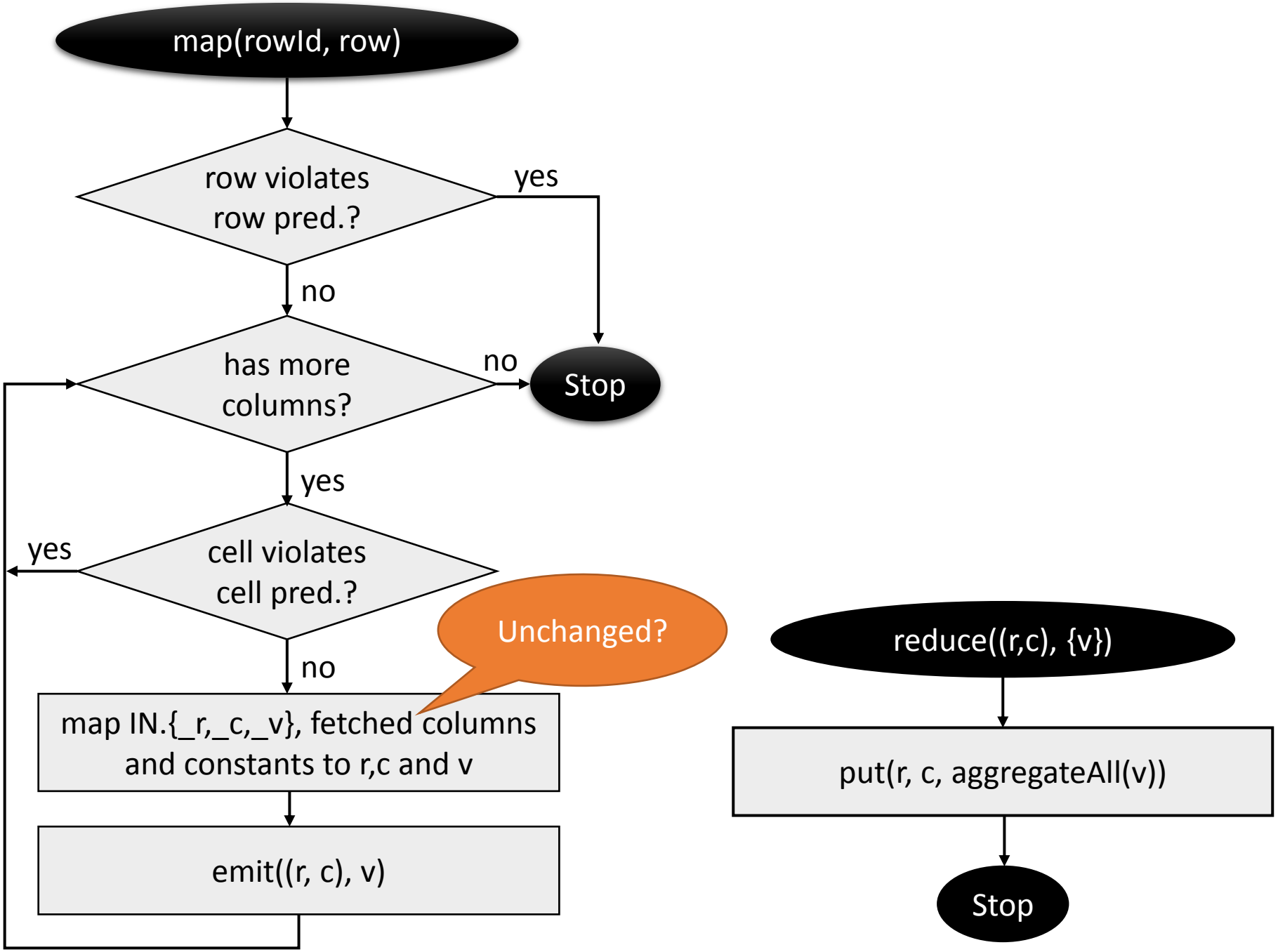
	born	cmpny	salary
Peter	1965	~~IBM~~ SAP	70k

-= 70k


map(rowId, row)
Just read Delta
(ts>ts‘)
row violates
row pred.?
yes
no
has more
columns?
no
Stop
yes
cell violates
cell pred.?
yes
no
map IN.{_r,_c,_v}, fetched columns
and constants to r,c and v
emit((r, c), v)
reduce((r,c), {v})
put(r, c, aggregateAll(v))
Stop

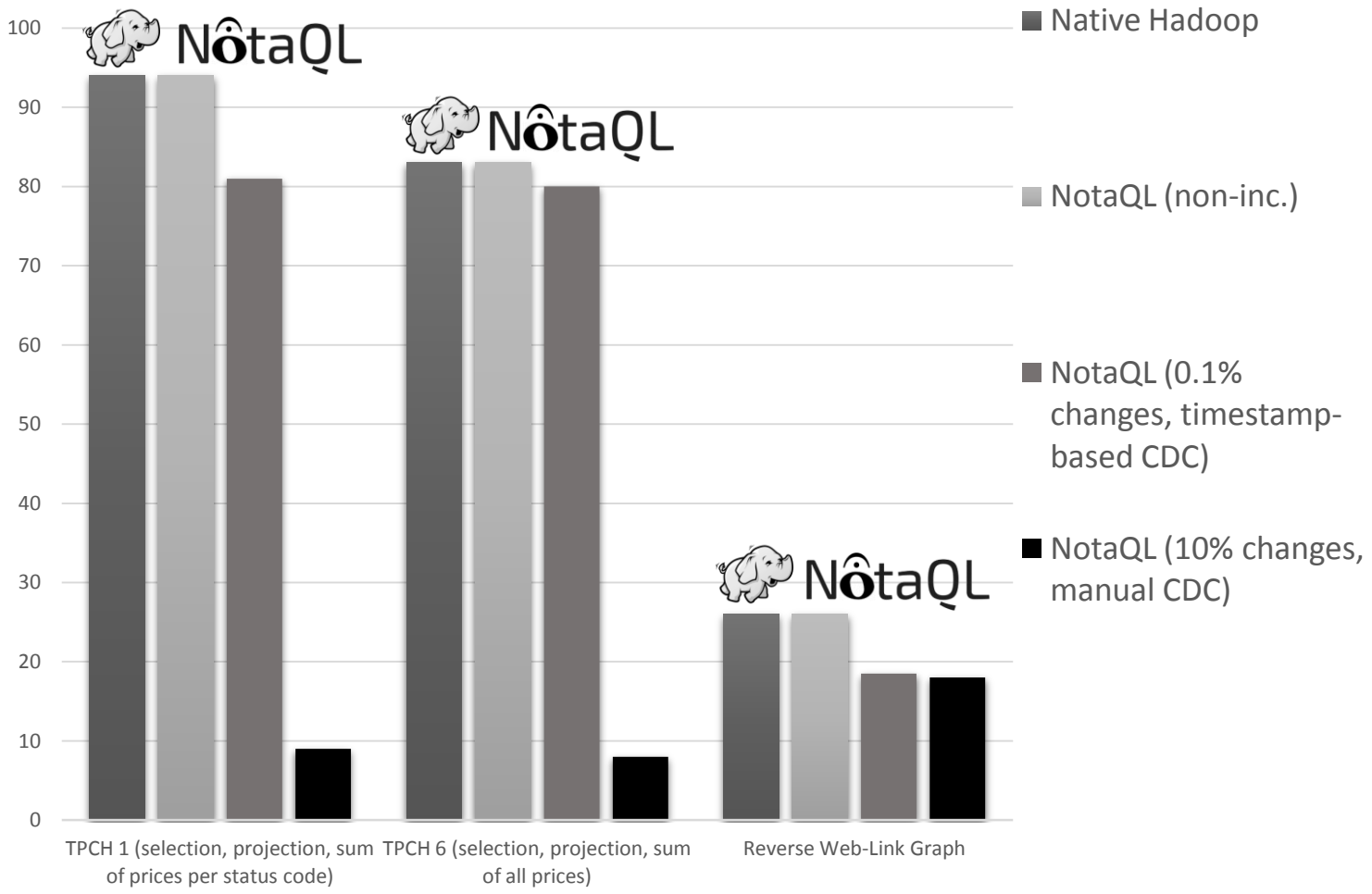

map(rowId, row)
…and the former Result
row violates row pred.?
yes
no
has more columns?
no
Stop
yes
cell violates cell pred.?
yes
no
map IN.{_r,_c,_v}, fetched columns and constants to r,c and v
emit((r, c), v)
reduce((r,c), {v})
put(r, c, aggregateAll(v))
Stop


map(rowId, row)
Was it satisfied on previous execution?
Yes? Invert v
row violates row pred.?
no
has more columns?
no
Stop
yes
yes
cell violates cell pred.?
no
map IN.{_r,_c,_v}, fetched columns and constants to r,c and v
emit((r, c), v)*
reduce((r,c), {v})
put(r, c, aggregateAll(v))
Stop


map(rowId, row)
row violates row pred.?
yes
no
has more columns?
no
Stop
yes
cell violates cell pred.?
yes
no
Unchanged?
map IN.{_r,_c,_v}, fetched columns and constants to r,c and v
emit((r, c), v)
reduce((r,c), {v})
put(r, c, aggregateAll(v))
Stop

Evaluation:

Performance


100
90
80
70
60
50
40
30
20
10
0
NotaQL
NotaQL
NotaQL
Native Hadoop
NotaQL (non-inc.)
NotaQL (0.1% changes, timestamp-based CDC)
NotaQL (10% changes, manual CDC)
TPCH 1 (selection, projection, sum of prices per status code)
TPCH 6 (selection, projection, sum of all prices)
Reverse Web-Link Graph

Conclusion

≠SQL

- Selection, Projection ✓
- Grouping, Aggregation ✓
- Schema-Flexible ✗
- Horizontal Aggregation ✗
- Metadata↔Data ✗
- Graph Processing ✗
- Text Processing ✗

Thank you!

$$
\begin{aligned}
\langle \text{NotaQL} \rangle &\models [\text{IN-FILTER: } \langle \text{predicate} \rangle,] \langle \text{rowspec} \rangle, \langle \text{cellspec} \rangle (, \langle \text{cellspec} \rangle) * [;] \\
\langle \text{rowspec} \rangle &\models \text{OUT.\_r} <\text{-} \langle \text{vdata} \rangle \\
\langle \text{cellspec} \rangle &\models \text{OUT.}(\langle \text{colname} \rangle \mid \$(\langle \text{input} \rangle)) <\text{-} (\langle \text{vdata} \rangle \mid \langle \text{aggfun} \rangle (\langle \text{vdata} \rangle) \\
\langle \text{input} \rangle &\models (\text{IN.\_r} \mid \text{IN.}[\langle \text{colfamily} \rangle :](\text{\_c} \mid \text{\_v}) \mid \text{IN.}\langle \text{colname} \rangle)[?(\langle \text{predicate} \rangle)] \\
\langle \text{vdata} \rangle &\models \langle \text{input} \rangle \mid \langle \text{const} \rangle \mid \langle \text{vdata} \rangle (+ \mid - \mid * \mid /) \langle \text{vdata} \rangle \\
\langle \text{aggfun} \rangle &\models \text{COUNT} \mid \text{SUM} \mid \text{MIN} \mid \text{MAX} \mid \text{AVG} \\
\langle \text{const} \rangle &\models \text{'}(A \ldots Z \mid a \ldots z \mid 0 \ldots 9) + \text{'} \mid (0 \ldots 9) + \\
\langle \text{colname} \rangle &\models [\langle \text{colfamily} \rangle :](A \ldots Z \mid a \ldots z \mid 0 \ldots 9) + \\
\langle \text{colfamily} \rangle &\models (A \ldots Z \mid a \ldots z \mid 0 \ldots 9) + \\
\langle \text{predicate} \rangle &\models (\langle \text{colname} \rangle \mid @ \mid \text{col\_count}([\langle \text{colfamily} \rangle])) [\langle \text{op} \rangle (\langle \text{colname} \rangle \mid \langle \text{const} \rangle] \\
&\quad \mid (\text{NOT} \mid !) \langle \text{predicate} \rangle \mid \langle \text{predicate} \rangle (\text{AND} \mid \text{OR}) \langle \text{predicate} \rangle \\
\langle \text{op} \rangle &\models = \mid ! = \mid < \mid <= \mid > \mid >=
\end{aligned}
$$