Sound and Efficient Language-Integrated Query

Maintaining the `ORDER`

Oleg Kiselyov   Tatsuya Katsushima

Tohoku University, Japan

APLAS 2017
November, 2017

Outline

- **Motivation**

  Core SQUR

  Core SQUR with Ranking

  Conclusions

Stackoverflow Motivation

"Query with `ORDER BY` in a `FROM` subquery produces unordered result. Is this a bug? Below is an example of this:

```
SELECT field1, field2
 FROM ( SELECT field1, field2 FROM table_1 ORDER BY field2) alias
```

returns a result set that is not necessarily ordered by field2."

`https://mariadb.com/kb/en/mariadb/why-is-order-by-in-a-from-subquery-ignored/`

Stackoverflow Motivation

"Query with `ORDER BY` in a `FROM` subquery produces unordered result. Is this a bug? Below is an example of this:

```
SELECT field1, field2
 FROM ( SELECT field1, field2 FROM table1 ORDER BY field2) alias
```

returns a result set that is not necessarily ordered by field2."

`https://mariadb.com/kb/en/mariadb/why-is-order-by-in-a-from-subquery-ignored/`

"A "table" (and subquery in the `FROM` clause too) is – according to the SQL standard – an unordered set of rows.... That's why the optimizer can ignore the `ORDER BY` clause that you have specified. In fact, SQL standard does not even allow the `ORDER BY` clause to appear in this subquery (we allow it, because `ORDER BY...LIMIT` changes the result, the set of rows, not only their order)."

Stackoverflow Motivation

"I am using an application (MapServer) that wraps SQL statements, so that the `ORDER BY` statement is in the inner query. E.g.

```
SELECT * FROM (SELECT ID, GEOM, Name FROM t ORDER BY Name) as tbl
```

The application has many different database drivers. I mainly use the MS SQL Server driver, and SQL Server 2008. This throws an error if an `ORDER BY` is found in a subquery."

`http://dba.stackexchange.com/questions/82930/database-implementations-of-order-by-in-a-subquery`

"However the same type of query when run in Postgres (9) and Oracle return results - with the order as defined in the subquery. In Postgres the query plan shows the results are sorted and the Postgres release notes include the item which implies subquery orders are used..."

`ORDER BY` in a subquery

- Not allowed at all

`ORDER BY` in a subquery

- Not allowed at all
  . . . unless LIMIT or TOP or FOR XML are also present

`ORDER BY` in a subquery

- Not allowed at all
  ... unless LIMIT or TOP or FOR XML are also present
- Allowed but ignored

`ORDER BY` in a subquery

- Not allowed at all
  ... unless LIMIT or TOP or FOR XML are also present
- Allowed but ignored
  ... unless LIMIT or TOP or FOR XML are also present

ORDER BY in a subquery

- Not allowed at all
  ... unless LIMIT or TOP or FOR XML are also present
- Allowed but ignored
  ... unless LIMIT or TOP or FOR XML are also present
  ... but not unless TOP 100%

`ORDER BY` in a subquery

- Not allowed at all
  ... unless LIMIT or TOP or FOR XML are also present
- Allowed but ignored
  ... unless LIMIT or TOP or FOR XML are also present
  ... but not unless TOP 100%
- Allowed and followed

`ORDER BY` in a subquery

- Not allowed at all
  ... unless LIMIT or TOP or FOR XML are also present
- Allowed but ignored
  ... unless LIMIT or TOP or FOR XML are also present
  ... but not unless TOP 100%
- Allowed and followed
  ... unless the subquery is too complex

depending on a database system or its version

Motivation 1

`ORDER BY` in a subquery is akin to undefined behavior in C

Just say NO!

Motivation 1

`ORDER BY` in a subquery is akin to undefined behavior in C

Just say NO!

Just write the perfect code: there will be no bugs

Motivation 1

`ORDER BY` in a subquery is akin to undefined behavior in C

Just say NO!

Just write the perfect code: there will be no bugs

But even then...

Stackoverflow Motivation

"I am using an application (MapServer) that wraps SQL statements, so that the `ORDER BY` statement is in the inner query. E.g.

```
SELECT * FROM (SELECT ID, GEOM, Name FROM t ORDER BY Name) as tbl
```

The application has many different database drivers. I mainly use the MS SQL Server driver, and SQL Server 2008. This throws an error if an `ORDER BY` is found in a subquery."

http://dba.stackexchange.com/questions/82930/database-implementations-of-order-by-in-a-subquery

Motivation 1

Why do people, and MapServer (and HRR, Opaleye, ...) keep writing `ORDER BY` in subqueries?

Motivation 1

Why do people, and MapServer (and HRR, Opaleye, . . .)
keep writing `ORDER BY` in subqueries?

Modularity, Reuse, Compositionality

Nested Relational Calculus

for (e $\leftarrow$ table employee) **where** e.wage>20 **yield** e

SELECT E.$*$ FROM employee **as** E WHERE E.wage $>$ 20

Language Integrated Query

```
for (e ← table employee) where e.wage>20 yield e
```

```
module Qe (S:SYM_SCHEMA) = struct open S
  let res = fun wage_thr →
      foreach (table t_employee) @@ fun e →
      where (wage e > wage_thr ) @@ fun () →
      yield e
end
```

Language Integrated Query

```
for (e ← table employee) for (d ← table department)
 where e.deptID = d.deptID yield <name=e.name, dep=d.name, wage=e.wage>
```

```
module Qd(S:SYM_SCHEMA) = struct open S
  let res = fun emp →
      foreach emp @@ fun e →
      foreach (table t_department) @@ fun d →
      where (deptID e = deptID d) @@ fun () →
      yield (emp_report (name e) (name d) (wage e))
end
```

Query Composition

```
Qd.res (Qe.res 20)
```

for (e ← **for**(e ← table employee) **where** e.wage>20 **yield** e)
for (d ← table department)
 where e.deptID = d.deptID **yield** <name=e.name, dep=d.name, wage=e.wage>

Query Composition

```
Qd.res (Qe.res 20)
```

```
for (e ← for(e ← table employee) where e.wage>20 yield e)
for (d ← table department)
 where e.deptID = d.deptID yield <name=e.name, dep=d.name, wage=e.wage>
```

```
SELECT E.name, D.name, E.wage
 FROM department D,
      (SELECT E.∗ FROM employee as E WHERE E.wage > 20) E
 WHERE D.deptID=E.deptID
```

Motivation 1

How `ORDER BY` hurts

The meaning of `ORDER BY` differs depending on:

- if attached to the top query
- if accompanied by `LIMIT`
- if appearing in a subquery

Motivation 1

How `ORDER BY` hurts

The meaning of `ORDER BY` differs depending on:

- if attached to the top query
- if accompanied by `LIMIT`
- if appearing in a subquery

Need a compositional semantics of `ORDER BY`

Query Normalization

for (e $\leftarrow$ **for**(e $\leftarrow$ table employee) **where** e.wage>20 **yield** e)
for (d $\leftarrow$ table department)
where e.deptID = d.deptID **yield** <name=e.name, dep=d.name, wage=e.wage>

for (e $\leftarrow$ table employee)
for (d $\leftarrow$ table department)
where e.deptID = d.deptID && e.wage>20
yield <name=e.name, dep=d.name, wage=e.wage>

```
SELECT E.name, D.name, E.wage
 FROM department D, employee E
 WHERE D.deptID=E.deptID AND E.wage > 20
```

Motivation 2

How to normalize language-integrated queries with ranking

Normalization is now a must: subqueries with ranking have no well-defined and portable semantics

Ranking Challenges

Distributivity laws of `UNION ALL`

$$
\begin{array}{lcl}
\textbf{for}\,(x \leftarrow e_1 \uplus e_2)\; e & \equiv & (\textbf{for}\,(x \leftarrow e_1)\; e) \;\uplus\; (\textbf{for}\,(x \leftarrow e_2)\; e) \\
\textbf{for}\,(x \leftarrow e)\; e_1 \uplus e_2 & \equiv & (\textbf{for}\,(x \leftarrow e)\; e_1) \;\uplus\; (\textbf{for}\,(x \leftarrow e)\; e_2) \\
\textbf{where}\; e\; e_1 \uplus e_2 & \equiv & (\textbf{where}\; e\; e_1) \;\uplus\; (\textbf{where}\; e\; e_2)
\end{array}
$$

$$
\begin{array}{l}
\textbf{ordering}\_\mathrm{wage}\;(\textbf{for}(x \leftarrow e_1 \uplus e_2)\; e) \;\not\equiv \\
\;(\textbf{ordering}\_\mathrm{wage}\;(\textbf{for}(x \leftarrow e_1)\; e)) \;\uplus\; (\textbf{ordering}\_\mathrm{wage}\;(\textbf{for}(x \leftarrow e_2)\; e))
\end{array}
$$

Motivation 2

How to normalize language-integrated queries with ranking

Normalization is now a must: subqueries with ranking have no well-defined and portable semantics

What are the equational laws for queries with ranking?

Motivations

- How to use `ORDER BY` in language-integrated queries and know what they mean?
- How to generate portable SQL from composed queries?

Motivations

- How to use `ORDER BY` in language-integrated queries and know what they mean?
- How to generate portable SQL from composed queries?

- What does ORDER BY actually mean?
- What are the equational laws for queries with ranking?

Motivations

- How to use `ORDER BY` in language-integrated queries and know what they mean?
- How to generate portable SQL from composed queries?

- What does ORDER BY actually mean?
- What are the equational laws for queries with ranking?

Denotational semantics made simple

Outline

Motivation

- **Core SQUR**

Core SQUR with Ranking

Conclusions

SQUR, formally

Variables	x,y,z. . .
Constants	c (integers, booleans, tables, etc.)
Numeric Literals	n, m
Record Labels	l
Effect Annotations	ϵ
Base Types	b ::= int | bool | string
Flat Types	t ::= b | <l:b,. . .>
Types	s ::= t | t bag^ϵ | t tbl
Type Environment	Γ ::= x:t, y:t tbl, . . .
Expressions	e ::= c | x | e + e | <l=e,. . .> | e.l | **for** (x←e) e | e ⊎ e | **where** e e | **yield** e | table e

SQUR, formally

Variables	x,y,z...
Constants	c (integers, booleans, tables, etc.)
Numeric Literals	n, m
Record Labels	l
Effect Annotations	ϵ
Base Types	b ::= int | bool | string
Flat Types	t ::= b | <l:b,...>
Types	s ::= t | t bag^ϵ | t tbl
Type Environment	Γ ::= x:t, y:t tbl, ...
Expressions	e ::= c | x | e + e | <l=e,...> | e.l | **for** (x←e) e | e $\uplus$ e | **where** e e | **yield** e | table e

There are no λ

Type System

$$\frac{}{\Gamma \vdash \mathsf{employee}\colon \langle \mathsf{name{:}string},\ \mathsf{deptID{:}int},\ \mathsf{wage{:}int} \rangle\ \mathsf{tbl}}\ \text{Const}$$

$$\frac{\Gamma \vdash \mathsf{e}\colon\ \mathsf{t}\ \ \mathsf{tbl}}{\Gamma \vdash \mathsf{table}\ \ \mathsf{e}\ :\ \mathsf{t}\ \ \mathsf{bag}\hat{}\phi}\ \text{Table}$$

$$\frac{\Gamma \vdash \mathsf{e}_1\colon\ \mathsf{t}\ \ \mathsf{bag}\hat{}\epsilon \qquad \Gamma \vdash \mathsf{e}_2\colon\ \mathsf{t}\ \mathsf{bag}\hat{}\epsilon}{\Gamma \vdash \mathsf{e}_1 \uplus \mathsf{e}_2\colon\ \mathsf{t}\ \ \mathsf{bag}\hat{}\epsilon}\ \text{UnionAll}$$

$$\frac{\Gamma \vdash \mathsf{e}_1\colon\ \mathsf{t}_1\ \ \mathsf{bag}\hat{}\phi \qquad \Gamma,\mathsf{x{:}t}_1 \vdash \mathsf{e}_2\colon\ \mathsf{t}_2\ \ \mathsf{bag}\hat{}\epsilon}{\Gamma \vdash \mathbf{for}(\mathsf{x}\leftarrow\mathsf{e}_1)\ \mathsf{e}_2\colon\ \mathsf{t}_2\ \ \mathsf{bag}\hat{}\epsilon}\ \text{For}$$

Denotational Semantics: Types

$$
\begin{aligned}
\mathcal{T}[\mathsf{int}] &= \mathbb{N} \\
\mathcal{T}[\mathsf{bool}] &= \{T, F\} \\[1ex]
\mathcal{T}[<\mathsf{l}_1{:}\mathsf{b}_1,\ldots,\mathsf{l}_n{:}\mathsf{b}_n>] &= \mathsf{l}_1{:}\mathcal{T}[\mathsf{b}_1] \times \cdots \times \mathsf{l}_n{:}\mathcal{T}[\mathsf{b}_n] \\
\mathcal{T}[\mathsf{t\ tbl}] &= \{\{\mathcal{T}[\mathsf{t}]\}\} \\
\mathcal{T}[\mathsf{t\ bag}] &= \{\{\mathcal{T}[\mathsf{t}]\}\} \\[1ex]
\mathcal{T}[\mathsf{x}_1{:}\mathsf{t}_1,\ldots,\mathsf{x}_n{:}\mathsf{t}_n] &= \mathsf{x}_1{:}\mathcal{T}[\mathsf{t}_1] \times \cdots \times \mathsf{x}_n{:}\mathcal{T}[\mathsf{t}_n]
\end{aligned}
$$

The very standard and conventional multiset semantics

Denotational Semantics: Values

$$
\begin{array}{lcl}
\mathcal{E}[\Gamma \vdash \mathsf{c:\ s}]\ \rho & \in & \mathcal{T}[\mathsf{s}] \\[4pt]
\mathcal{E}[\Gamma \vdash \mathsf{bag\_empty:\ t\ bag}]\ \rho & = & \{\{\}\} \\
\mathcal{E}[\Gamma \vdash \mathsf{x:\ t}]\ \rho & = & \rho.\mathsf{x} \\
\mathcal{E}[\Gamma \vdash \mathsf{e}_1\ \texttt{+}\ \mathsf{e}_2\mathsf{:\ int}]\ \rho & = & \\
\quad \mathcal{E}[\Gamma \vdash \mathsf{e}_1\mathsf{:\ int}]\rho + \mathcal{E}[\Gamma \vdash \mathsf{e}_2\mathsf{:\ int}]\rho & &
\end{array}
$$

Denotational Semantics: Values

$$\mathcal{E}[\Gamma \vdash \mathsf{e}_1 \uplus \mathsf{e}_2\text{: } \mathsf{t\ bag}]\ \rho \quad = \quad \mathcal{E}[\Gamma \vdash \mathsf{e}_1\text{: } \mathsf{t\ bag}]\rho \cup \mathcal{E}[\Gamma \vdash \mathsf{e}_2\text{: } \mathsf{t\ bag}]\rho$$

$$\mathcal{E}[\Gamma \vdash \textbf{yield}\ \mathsf{e}\text{: } \mathsf{t\ bag}]\ \rho \quad = \quad \{\{\ \mathcal{E}[\Gamma \vdash \mathsf{e}\text{: } \mathsf{t}]\rho\ \}\}$$

$$\mathcal{E}[\Gamma \vdash \textbf{where}\ \mathsf{e}_1\ \mathsf{e}\text{: } \mathsf{t\ bag}]\ \rho \quad = \quad \textbf{if}\ \mathcal{E}[\Gamma \vdash \mathsf{e}_1\text{: } \mathsf{bool}]\rho\ \textbf{then}\ \mathcal{E}[\Gamma \vdash \mathsf{e}\text{: } \mathsf{t\ bag}]\rho\ \textbf{else}\ \{\{\}\}$$

$$\mathcal{E}[\Gamma \vdash \textbf{for}(\mathsf{x} \leftarrow \mathsf{e}_1)\ \mathsf{e}\text{: } \mathsf{t\ bag}]\ \rho \quad = \quad \bigcup\{\{\ \mathcal{E}[\Gamma, \mathsf{x}\text{:}\mathsf{t}_1 \vdash \mathsf{e}\text{: } \mathsf{t\ bag}]\ (\rho \times \mathsf{x} : x') \mid x' \leftarrow \mathcal{E}[\Gamma \vdash \mathsf{e}_1\text{: } \mathsf{t}_1\ \mathsf{bag}]\rho\ \}\}$$

$\textbf{for}(\mathsf{x} \leftarrow \mathsf{e}_1)\ \mathsf{e}$ is truly a bag (multiset) comprehension

Application: Distributivity Laws

Distributivity laws of `UNION ALL`

$$
\begin{array}{lcl}
\textbf{for}(x \leftarrow e_1 \uplus e_2)\ e & \equiv & (\textbf{for}(x \leftarrow e_1)\ e) \uplus (\textbf{for}(x \leftarrow e_2)\ e) \\
\textbf{for}(x \leftarrow e)\ e_1 \uplus e_2 & \equiv & (\textbf{for}(x \leftarrow e)\ e_1) \uplus (\textbf{for}(x \leftarrow e)\ e_2) \\
\textbf{where}\ e\ e_1 \uplus e_2 & \equiv & (\textbf{where}\ e\ e_1) \uplus (\textbf{where}\ e\ e_2)
\end{array}
$$

Application: NBE

Normalization by Rewriting

- Syntactic: term re-writing
- Non-deterministic
- Need to assure (prove) confluence and termination
- Normal form emerges as the result
  (hope it is translatable to SQL)

Normalization by Evaluation

- Normal form is designed first, to be translatable to SQL
- Semantic: non-standard evaluation
- Deterministic and terminating

NBE

$$
\begin{aligned}
\mathcal{T}^n[\mathsf{int}] &= \mathbb{N} \oplus \mathbb{E}_{\mathsf{int}} \\
\mathcal{E}^n[\Gamma \vdash \mathsf{0}\colon \mathsf{int}]\ \rho &= 0 \\[1ex]
\mathcal{E}^n[\Gamma \vdash \mathsf{e}_1 \ \mathsf{+}\ \mathsf{e}_2\colon \mathsf{int}]\ \rho &= \mathit{add}\ (\mathcal{E}^n[\Gamma \vdash \mathsf{e}_1\colon \mathsf{int}]\ \rho)\ (\mathcal{E}^n[\Gamma \vdash \mathsf{e}_2\colon \mathsf{int}]\ \rho) \\
\quad \mathit{add}\ 0\ \mathrm{x} &= \mathrm{x} \\
\quad \mathit{add}\ \mathrm{x}\ 0 &= \mathrm{x} \\
\quad \mathit{add}\ n\ m &= n + m \qquad n, m \in \mathbb{N} \\
\quad \mathit{add}\ \mathrm{x}\ \mathrm{y} &= \mathit{inr}\ (\mathcal{I}[\mathsf{x}]\ \mathsf{+}\ \mathcal{I}[\mathsf{y}]) \\[1ex]
\mathcal{I}[-]\colon & \quad \mathcal{T}^n[\mathsf{s}] \to \mathbb{E}_{\mathsf{s}} \\
\mathcal{I}[0] &= \mathsf{0} \\
\mathcal{I}[\mathit{inr}\ \mathsf{e}] &= \mathsf{e}
\end{aligned}
$$

In **for** ($\mathsf{x} \leftarrow \mathsf{table}\ \mathsf{t}_1$) **yield** 1+2+x, $\mathcal{E}^n[\mathsf{1+2+x}]$ is inr (3+x)

NBE: Bags

$$\mathcal{T}^n[\mathsf{t\ bag}] = \{\{\ \mathtt{fors}(\mathsf{x}{\leftarrow}\mathbb{M}\ldots)\ \mathtt{whr}\ \mathcal{T}^n[\mathsf{bool}]\ \mathtt{yld}\ \mathcal{T}^n[\mathsf{t}]\ \}\}$$

$$\mathcal{E}^n[\Gamma \vdash \mathsf{bag\_empty}\colon \mathsf{t\ bag}]\ \rho = \{\{\}\}$$

$$\mathcal{E}^n[\Gamma \vdash \mathsf{e}_1 \uplus \mathsf{e}_2\colon \mathsf{t\ bag}]\ \rho = \mathcal{E}^n[\Gamma \vdash \mathsf{e}_1\colon \mathsf{t\ bag}]\rho \cup \mathcal{E}^n[\Gamma \vdash \mathsf{e}_2\colon \mathsf{t\ bag}]\rho$$

NBE: Bags

$$\mathcal{E}^n[\Gamma \vdash \textbf{yield}\ \mathsf{e}\text{: } \mathsf{t\ bag}]\ \rho = \{\!\{\ \texttt{fors}\ ()\ \texttt{whr}\ T\ \texttt{yld}\ \mathcal{E}^n[\Gamma \vdash \mathsf{e}\text{: } \mathsf{t}]\rho\ \}\!\}$$

$$\mathcal{E}^n[\Gamma \vdash \mathsf{table\ m}\text{: } \mathsf{t\ bag}]\ \rho = \{\!\{\ \texttt{fors}\ (\mathsf{u}\!\leftarrow\!\mathsf{m})\ \texttt{whr}\ T\ \texttt{yld}\ \mathsf{u}\ \}\!\}\ \text{and } \mathsf{u} \text{ is fresh}$$

$$\mathcal{E}^n[\Gamma \vdash \textbf{where}\ \mathsf{e}_1\ \mathsf{e}\text{: } \mathsf{t\ bag}]\ \rho = \mathit{where}'\ (\mathcal{E}^n[\Gamma \vdash \mathsf{e}_1\text{: } \mathsf{bool}]\rho)\ (\mathcal{E}^n[\Gamma \vdash \mathsf{e}\text{: } \mathsf{t\ bag}]\rho)\ \text{where}$$

$$\begin{aligned}
\mathit{where}'\ T\ xs &= xs \\
\mathit{where}'\ F\ xs &= \{\!\{\}\!\} \\
\mathit{where}'\ t\ xs &= \{\!\{\ \texttt{fors}\ (\mathsf{x}\!\leftarrow\!\mathsf{m}\ldots)\ \texttt{whr}\ w \wedge t\ \texttt{yld}\ y \\
&\qquad \mid\ \texttt{fors}\ (\mathsf{x}\!\leftarrow\!\mathsf{m}\ldots)\ \texttt{whr}\ w\ \texttt{yld}\ y \leftarrow xs\}\!\}
\end{aligned}$$

NBE: Bags

$$
\begin{aligned}
&\mathcal{E}^n[\Gamma \vdash \textbf{for}(\mathsf{x}\leftarrow \mathsf{e}_1)\ \mathsf{e}\text{: t bag}]\ \rho \quad = \\
&\quad \{\{\ \texttt{fors}\ (\mathsf{x'}\leftarrow \mathsf{m'},\ldots \mathsf{x''}\leftarrow \mathsf{m''},\ldots)\ \texttt{whr}\ w' \wedge w''\ \texttt{yld}\ y'' \mid \\
&\qquad \texttt{fors}\ (\mathsf{x'}\leftarrow \mathsf{m'}\ldots)\ \texttt{whr}\ w'\ \texttt{yld}\ y' \leftarrow \mathcal{E}^n[\Gamma \vdash \mathsf{e}_1\text{: }\mathsf{t}_1\text{ bag}]\rho, \\
&\qquad \texttt{fors}\ (\mathsf{x''}\leftarrow \mathsf{m''}\ldots)\ \texttt{whr}\ w''\ \texttt{yld}\ y'' \leftarrow \\
&\qquad\qquad \mathcal{E}^n[\Gamma,\mathsf{x}\text{:}\mathsf{t}_1 \vdash \mathsf{e}\text{: t bag}](\rho \times \mathsf{x}\text{:}y')\ \}\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{I}[\{\{\}\}] \quad = \text{bag\_empty} \\
&\mathcal{I}[xs] \quad = \\
&\quad \biguplus \{\{\ \textbf{for}(\mathsf{x}\leftarrow\text{table m})\ \ldots\ \textbf{where}\ \mathcal{I}[w]\ \textbf{yield}\ \mathcal{I}[y] \mid \\
&\qquad \texttt{fors}(\mathsf{x}\leftarrow \mathsf{m}\ldots)\ \texttt{whr}\ w\ \texttt{yld}\ y \leftarrow xs\}\}
\end{aligned}
$$

(the **where** clause is omitted if w is T)

Query Normalization

query:

```
for (e ← for(e ← table employee) where e.wage>20 yield e)
for (d ← table department)
 where e.deptID = d.deptID yield <name=e.name, dep=d.name, wage=e.wage>
```

$\mathcal{E}^n$[⊢ query : <name,dep,wage> bag]:

```
{{ fors(e←employee d←department)
   whr (e.deptID = d.deptID && e.wage>20)
   yld <name=e.name, dep=d.name, wage=e.wage>
}}
```

$\mathcal{I}$[$\mathcal{E}^n$[⊢ query : <name,dep,wage> bag]]:

```
for (e ← table employee)
for (d ← table department)
 where e.deptID = d.deptID && e.wage>20
 yield <name=e.name, dep=d.name, wage=e.wage>
```

Formal Properties

Theorem (Type Preservation)

For all $\Gamma \vdash \mathsf{e}{:}\mathsf{s}$ *and* $\rho \in \mathcal{T}^n[\![\Gamma]\!]$, *it holds* $\Gamma' \vdash \mathcal{I}[\![\mathcal{E}^n[\![\mathsf{e}]\!]\rho]\!]$, *where* Γ' *lists the variables in the domain of* ρ *and their types.*

Theorem (Soundness of NBE)

For all SQUR *expressions* $\Gamma \vdash \mathsf{e}{:}\mathsf{s}$, *and environments* ρ *and* ρ' *of appropriate types,* $\mathcal{E}[\![\, \mathcal{I}[\![\mathcal{E}^n[\![\mathsf{e}]\!]\rho]\!] \,]\!]\rho'$ *is equal to* $\mathcal{E}[\![\mathsf{e}]\!](\mathcal{E}[\![\mathcal{I}[\![\rho]\!]]\!]\rho')$.

Normalization, Formally

Definition (Normal form)

We call $\mathcal{I}[\mathcal{E}^n[\mathsf{e}]\mathsf{<>}]$ the normal form $\mathcal{N}[\mathsf{e}]$ of a closed term e

Theorem (Correctness of normal form)

If e is a closed term of the type s, then

(a) $\mathcal{N}[e]$ *exists*

(b) $\vdash \mathcal{N}[e]{:}s$

(c) $\mathcal{N}[\mathcal{N}[e]] = \mathcal{N}[e]$

(d) $\mathcal{E}[e] = \mathcal{E}[\mathcal{N}[e]]$

SQL Translation

```
{{ fors(e←employee d←department)
   whr (e.deptID = d.deptID && e.wage>20)
   yld <name=e.name, dep=d.name, wage=e.wage>
}}
```

```
for  (e ← table  employee)
for  (d ← table  department)
 where e.deptID = d.deptID && e.wage>20
  yield  <name=e.name, dep=d.name, wage=e.wage>
```

```
SELECT E.name, D.name, E.wage
 FROM department D, employee E
 WHERE D.deptID=E.deptID AND E.wage > 20
```

Outline

Ordering and Subranging

```
for (e ← table employee) where e.wage>20
  ordering_wage e.wage yield e
```

```
SELECT E.* FROM employee as E
WHERE E.wage > 20 ORDER BY E.wage
```

Ordering and Subranging

for (e $\leftarrow$ table employee) **where** e.wage$>$20
ordering_wage e.wage **yield** e

SELECT E.$*$ FROM employee **as** E
WHERE E.wage $>$ 20 ORDER BY E.wage

for (e $\leftarrow$ table employee) **where** e.wage$>$20
limit (3,1) **ordering**_wage e.wage **yield** e

SELECT E.$*$ FROM employee **as** E
WHERE E.wage $>$ 20 ORDER BY E.wage
LIMIT 3 OFFSET 1

SQUR with ranking

Ordering Effects	o:[olabel,...], l:(n,m)
Ordering Labels	owage,...
Expressions	e +:= **ordering**_wage e_1 e | **limit** (n,m) e | **let** table x = e **in** e

SQUR with ranking: Types

$$\frac{\Gamma \vdash \mathsf{e}_1\colon\ \mathsf{int} \qquad \Gamma \vdash \mathsf{e}\colon \mathsf{t}\ \mathsf{bag}\hat{}\epsilon \qquad \epsilon \subseteq \{\mathsf{o}\colon[\mathsf{lb},\ldots]\}}{\Gamma \vdash \textbf{ordering}\_\mathsf{wage}\ \mathsf{e}_1\ \mathsf{e}\ :\ \mathsf{t}\ \ \mathsf{bag}\hat{}\{\mathsf{o}\colon[\mathsf{owage},\mathsf{lb},\ldots]\}}\ \text{Ordering}$$

$$\frac{\Gamma \vdash \mathsf{e}\colon\ \mathsf{t}\ \ \mathsf{bag}\hat{}\epsilon \qquad \epsilon = \{\mathsf{o}\colon[\mathsf{lb},\ldots]\}}{\Gamma \vdash \textbf{limit}\ \ (\mathsf{n},\mathsf{m})\ \mathsf{e}\ :\ \mathsf{t}\ \ \mathsf{bag}\hat{}(\epsilon \cup \{\mathsf{l}\colon(\mathsf{n},\mathsf{m})\})}\ \text{Limit}$$

$$\frac{\Gamma \vdash \mathsf{e}_1\colon\ \mathsf{t}_1\ \ \mathsf{bag}\hat{}\epsilon_1 \qquad \epsilon_1 \subseteq \{\mathsf{o}\colon[\mathsf{lb},\ldots]\} \qquad \Gamma,\mathsf{x}\colon\mathsf{t}_1 \vdash \mathsf{e}_2\colon\ \mathsf{t}_2\ \mathsf{bag}\hat{}\epsilon}{\Gamma \vdash \textbf{for}(\mathsf{x}\leftarrow\mathsf{e}_1)\ \mathsf{e}_2\colon\ \ \mathsf{t}_2\ \ \mathsf{bag}\hat{}\epsilon}\ \text{For}$$

$$\frac{\vdash \mathsf{e}_1\colon\ \mathsf{t}_1\ \ \mathsf{bag}\hat{}\epsilon_1 \qquad \Gamma,\mathsf{y}\colon\mathsf{t}_1\ \mathsf{tbl} \vdash \mathsf{e}_2\colon\ \mathsf{t}_2\ \mathsf{bag}\hat{}\epsilon}{\Gamma \vdash \textbf{let}\ \ \mathsf{table}\ \ \mathsf{y}{=}\mathsf{e}_1\ \textbf{in}\ \mathsf{e}_2\colon\ \ \mathsf{t}_2\ \ \mathsf{bag}\hat{}\epsilon}\ \text{Let}$$

Denotation of Ranking

$$\mathcal{T}[\text{t bag\^{}}\phi] = \{\{\ \mathcal{T}[\text{t}]\ \}\}$$

$$\mathcal{T}[\text{t bag\^{}}\{\text{o:[lb,}\ldots]\}] = \{\{\ \mathcal{T}[\text{t}] \times \text{o:}(\mathbb{N} \times \ldots)\ \}\}$$

$$\mathcal{T}[\text{t bag\^{}}\{\text{o:[lb,}\ldots\text{],l:(n,m)}\}] = \\ \{\{\ \mathcal{T}[\text{t}] \times \text{o:}(\mathbb{N} \times \ldots) \times \text{l:}(\mathbb{N} \times \mathbb{N})\ \}\}$$

$$\mathcal{E}[\Gamma \vdash \textbf{for}(\text{x} \leftarrow \text{e}_1)\ \text{e: t bag\^{}}\epsilon]\ \rho = \\ \bigcup \{\{\ \mathcal{E}[\Gamma,\text{x:t}_1 \vdash \text{e: t bag\^{}}\epsilon]\ (\rho \times \text{x} : x')\ | \\ x' \times \text{o:\_} \leftarrow \mathcal{E}[\Gamma \vdash \text{e}_1\text{: t}_1\text{ bag\^{}}\epsilon']\rho\ \}\}$$

Denotation of Ranking

$$\mathcal{E}[\Gamma \vdash \textbf{ordering\_}\mathsf{lb}\ \mathsf{e}_1\ \mathsf{e{:}\ t\ bag\hat{}}\{\mathsf{o{:}[lb]}\}]\ \rho =$$
$$\{\{\ x \times \mathsf{o{:}}[\mathcal{E}[\mathsf{e}_1]\rho]\ |\ \ x \leftarrow \mathcal{E}[\Gamma \vdash \mathsf{e{:}\ t\ bag\hat{}}\phi]\rho\ \}\}$$

$$\mathcal{E}[\Gamma \vdash \textbf{ordering\_}\mathsf{lb}\ \mathsf{e}_1\ \mathsf{e{:}\ t\ bag\hat{}}\epsilon]\ \rho =$$
$$\{\{\ x \times \mathsf{o{:}}[\mathcal{E}[\mathsf{e}_1]\rho,\mathsf{lb'},\ldots]\ |$$
$$x \times \mathsf{o{:}[lb',\ldots]} \leftarrow \mathcal{E}[\Gamma \vdash \mathsf{e{:}\ t\ bag\hat{}}\epsilon_1]\rho\ \}\}$$
$$\text{where } \epsilon_1{=}\{\mathsf{o{:}[lb',\ldots]}\} \text{ and } \epsilon{=}\{\mathsf{o{:}[lb,lb',\ldots]}\}$$

$$\mathcal{E}[\Gamma \vdash \textbf{limit}\ \mathsf{(n,m)}\ \mathsf{e{:}\ t\ bag\hat{}}\{\epsilon\ \mathsf{U}\ \mathsf{l{:}(n,m)}\}]\ \rho =$$
$$\{\{\ x \times \mathsf{l{:}(n,m)}\ |\ \ x \leftarrow \mathcal{E}[\Gamma \vdash \mathsf{e{:}\ t\ bag\hat{}}\epsilon]\rho\ \}\}$$

$$\mathcal{E}[\Gamma \vdash \textbf{let}\ \mathsf{table\ y{=}e_1}\ \textbf{in}\ \mathsf{e{:}\ t\ bag\hat{}}\epsilon]\ \rho =$$
$$\mathcal{E}[\Gamma,\mathsf{y{:}t_1\ tbl} \vdash \mathsf{e{:}\ t\ bag\hat{}}\epsilon]\ (\rho \times \mathsf{y}{:}\mathcal{M}[\vdash \mathsf{e}_1{:}\ \mathsf{t}_1\ \mathsf{bag\hat{}}\epsilon_1])$$

Denotation of Ranking

$$\mathcal{M}[\vdash \mathsf{e\colon t\ bag}\hat{}\epsilon] \quad \in \{\ \mathsf{sequence}[\mathcal{T}[\mathsf{t}]]\ \}$$

$$\mathcal{M}[\vdash \mathsf{e\colon t\ bag}\hat{}\phi] \quad = \mathcal{E}[\vdash \mathsf{e\colon t\ bag}\hat{}\phi]\mathsf{<>}$$

$$\mathcal{M}[\vdash \mathsf{e\colon t\ bag}\hat{}\epsilon] \quad =$$

$$\quad \mathit{subrange}\ (\mathsf{n,m}) \circ \mathit{sort}\ \mathsf{keys}$$

$$\qquad \{\{\ x \mid\ x \times \mathsf{o{:}keys} \times \mathsf{l{:}(n,m)} \leftarrow \mathcal{E}[\vdash \mathsf{e\colon t\ bag}\hat{}\epsilon]\mathsf{<>}\ \}\}$$

(no subranging if the l annotation is absent)

Application: Distributivity Laws

`UNION ALL` is associative and symmetric
Furthermore,

Theorem (Distributive Equational Laws of UNION ALL)

$$
\begin{array}{lcll}
\textbf{for}\,(x \leftarrow e_1 \uplus e_2)\; e & \equiv & (\textbf{for}\,(x{\leftarrow}e_1)\; e) & \uplus\; (\textbf{for}\,(x{\leftarrow}e_2)\; e) \\
\textbf{for}\,(x \leftarrow e)\; e_1 \uplus e_2 & \equiv & (\textbf{for}\,(x{\leftarrow}e)\; e_1) & \uplus\; (\textbf{for}\,(x{\leftarrow}e)\; e_2) \\
\textbf{where}\; e\; e_1 \uplus e_2 & \equiv & (\textbf{where}\; e\; e_1) & \uplus\; (\textbf{where}\; e\; e_2) \\
\textbf{ordering\_}\mathit{lb}\; e\; (e_1 \uplus e_2) & \equiv & (\textbf{ordering\_}\mathit{lb}\; e\; e_1) & \uplus\; (\textbf{ordering\_}\mathit{lb}\; e\; e_2) \\
\textbf{limit}\;\; (n, m)\; (e_1 \uplus e_2) & \equiv & (\textbf{limit}\;\; (n, m)\; e_1) & \uplus\; (\textbf{limit}\;\; (n, m)\; e_2)
\end{array}
$$

Sample Normalization

```
for  (e ←
   for (e ← table  employee) where e.wage>20
    ordering_wage e.wage yield  e)
 for  (d ← table  department)
  where e.deptID = d.deptID ordering_dept d.deptID
  yield  <name=e.name, dep=d.name, wage=e.wage>
```

Sample Normalization

```
for (e ←
   for(e ← table employee) where e.wage>20
    ordering_wage e.wage yield e)
 for (d ← table department)
  where e.deptID = d.deptID ordering_dept d.deptID
   yield <name=e.name, dep=d.name, wage=e.wage>
```

```
for (e ← table employee)
for (d ← table department)
where e.deptID = d.deptID && e.wage>20
ordering_dept d.deptID
yield <name=e.name, dep=d.name, wage=e.wage>
```

```
SELECT E.name, D.name, E.wage
FROM employee as E, department as D
WHERE E.deptID = D.deptID AND E.wage > 20
ORDER BY D.deptID
```

Sample Normalization

```
let table t =
  for (e ← table employee) where e.wage>20
   limit (3,1) ordering_wage e.wage yield e
in
for (e ← table t) for (d ← table department)
where e.deptID = d.deptID ordering_dept d.deptID
yield <name=e.name, dep=d.name, wage=e.wage>
```

```
WITH t8 AS (SELECT E.* FROM employee as E
              WHERE E.wage > 20 ORDER BY E.wage LIMIT 3 OFFSET 1)
SELECT t9.name, t7.name, t9.wage FROM department AS t7, t8 AS t9
WHERE t9.deptID = t7.deptID ORDER BY t7.deptID
```

Outline

Motivation

Core SQUR

Core SQUR with Ranking

▶ **Conclusions**

Conclusions

The first compositional, denotational treatment of `ORDER BY` and `LIMIT...OFFSET`

application for optimizing composed queries to yield efficient and *portable* SQL

- new calculus SQUR with ranking, the sound type system and denotational semantics
- Equational laws
  UNION ALL is still distributive and symmetric, even with ranking
- Normalization-by-evaluation
- Ranking as an effect

`http://okmij.org/ftp/meta-programming/Sqr/`