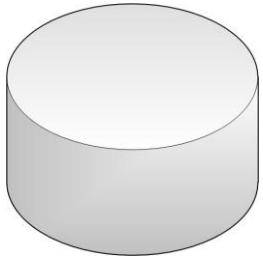

# *Thesis defense*

# Type safe integration of query languages into Scala

Christopher Vogt, 06.08.2011

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

RWTH AACHEN
UNIVERSITY

# Feature Comparison

| | SQL | Scala Query | SQueryl | Microsoft's LINQ-to-SQL on .NET | **Scala Integrated Query** |
|---|---|---|---|---|---|
| Queries | ✔ | ✔ | ✔ | ✔ | ✔ |
| Type Safety | | ✔ | ✔ | ✔ | ✔ |
| Nesting | | | | ✔ | ✔ |
| **Avalanche Safe Nesting** | | | | | ✔ |
| **Backend Specific Type Safety** | | | | | ✔ |

# Thesis content

**Formal translation steps**

- Complete round-trip:
  Scala -> Ferry -> Relational Algebra ->
  SQL:99 -> execution -> Scala results
- Avalanche Safety based on Ferry
- Nested tuples (extension of Ferry data model)

**Prototype implementation**

- Shows how type safety is achieved for different aspects
- Shows how translation steps can be implemented
- Backend specific type-safety due to modularity of light-weight modular staging

# Content of this talk

**Complete round-trip for an example query**

- Scala -> Ferry -> Relational Algebra -> SQL:99 -> execution -> results
- Nested tuples
- Avalanche Safety

# AVALANCHE SAFETY OVERVIEW

# Avalanche Safety

Number of queries

- = number of list constructors
- only depends on result type
- does not depend on database size

# How many SQL Queries?

**for nation <- nations**

**( nation, customers of this nation )**

**Avalanche Safe**

| | SQL | Scala Query | SQueryl | Microsoft's LINQ-to-SQL on .NET | **Scala Integrated Query** |
|---|---|---|---|---|---|
| **straight-forward** | **n + 1** | **n + 1** | **n + 1** | **n + 1** | **2** |
| **manually tweaked** | **2** | **2** | **2** | **2** | |

**n** = nations.length

# Translation at a glance

```
nation.map( n =>
    ( n,  customer.withFilter(_.nationkey == n.nationkey) )
) . fromdb : List[ ( Nation, List[Customer]  ) ]
```

**Query 1**

| … | name | inner |
|---|---|---|
| … | ARGENTINA | **1** |
| … | BRAZIL | **2** |
| .. | … | … |
| .. | GERMANY | **7** |
| .. | … | … |

**Query 2**

| outer | name | … |
|---|---|---|
| **1** | Miguel | … |
| … | … | … |
| **7** | Martin | … |
| **7** | Tiark | … |
| … | … | … |

# TRANSLATION STEPS

# Lifting based on LMS

- Scala representation of database schema
- overloads of map, withFilter, operators, etc. return AST instead of results

# Schema integration

```
val nation = Table[Nation](
        new Schema{
            val nationkey      = Column[Int]( "nationkey" )
            val name           = Column[String]( "name" )
})
            val customer = Table[Customer](
                new Schema{
                    val custkey        = Column[Int]( "id" )
                    val name           = Column[String]( "name" )
                    val nationkey      = Column[String]( "workgroup_id" )
            })

nation.map( n =>
    ( n,  customer.withFilter(_.nationkey == n.nationkey) )
)
```

*Schemas are auto-generated*

# Scala -> Ferry

```
nation.map( n =>
    ( n,   customer.withFilter(_.nationkey == n.nationkey) )
)
```

flattening of nested tuples


```
for n in table nation
    return ( n.nationkey, n.name,
            for c in table customer where c.nationkey == n.nationkey
            return (c.id, c.name, c.nationkey)
    )
```

# Ferry -> Relational Algebra

- avalanche safe, relational encoding of nested comprehensions
  - outermost list -> one query, one row per element
  - tuple -> row
  - atomic types -> columns
  - inner list -> surrogate key column + extra query

=> tree of relational queries

# Ferry -> Relational Algebra -> SQL

```
for n in table nation
    return ( n.nationkey, n.name,
        for c in table customer where c.nationkey == n.nationkey
        return (c.custkey, c.name, c.nationkey)
    )
```

Ferry to SQL

```
SELECT nationkey, name, ROW_NUMBER() AS inner FROM nation
```

**tree of relational queries**

```
WITH n AS
        SELECT nationkey, name, ROW_NUMBER() AS outer FROM nation
SELECT n.outer, c.custkey, c.name, c.nationkey
WHERE c.nationkey = n.nationkey
FROM n, customer c
```

# Relational Results

```
SELECT nationkey, name, ROW_NUMBER() AS inner FROM nation
```

```
WITH n AS
        SELECT nationkey, name, ROW_NUMBER() AS outer FROM nation
SELECT n.outer, c.custkey, c.name, c.nationkey
WHERE c.nationkey = n.nationkey
FROM n, customer c
```

| nationkey | name | inner |
|---|---|---|
| 1 | ARGENTINA | **1** |
| 2 | BRAZIL | **2** |
| … | … | … |
| 7 | GERMANY | **7** |
| … | … | … |

| outer | custkey | name | nationkey |
|---|---|---|---|
| **1** | 2 | Miguel | 1 |
| … | … | … | … |
| **7** | 1 | Martin | 7 |
| **7** | 3 | Tiark | 7 |
| … | … | … | … |

# Query 1 in detail

```
SELECT nationkey, name, ROW_NUMBER() AS inner FROM nation
```

| nationkey | name | inner |
|---|---|---|
| 1 | ARGENTINA | **1** |
| 2 | BRAZIL | **2** |
| … | … | **…** |
| 7 | GERMANY | **7** |
| … | … | **…** |

# Query 2 in detail

```
WITH n AS
        SELECT nationkey, name, ROW_NUMBER() AS outer FROM nation
SELECT n.outer, c.custkey, c.name, c.nationkey
WHERE c.nationkey = n.nationkey
FROM n, customer c
```

| nationkey | name | outer |
|---|---|---|
| 1 | ARGENTINA | **1** |
| 2 | BRAZIL | **2** |
| … | … | **…** |
| 7 | GERMANY | **7** |
| … | … | **…** |

| custkey | name | nationkey |
|---|---|---|
| 1 | Martin | 7 |
| 2 | Miguel | 1 |
| 3 | Tiark | 7 |
| … | … | … |

# Query 2 in detail

```
WITH n AS
        SELECT nationkey, name, ROW_NUMBER() AS outer FROM nation
SELECT n.outer, c.custkey, c.name, c.nationkey
WHERE c.nationkey = n.nationkey
FROM n, customer c
```

CARTESIAN PRODUCT

| nationkey | name | outer | custkey | name | nationkey |
|---|---|---|---|---|---|
| 1 | ARGENTINA | 1 | 1 | Martin | 7 |
| 1 | ARGENTINA | 1 | 2 | Miguel | 1 |
| … | … | … | … | … | … |
| 2 | BRAZIL | 2 | 1 | Martin | 7 |
| 2 | BRAZIL | 2 | 2 | Miguel | 1 |
| … | … | … | … | … | … |

# Query 2 in detail

```
WITH n AS
        SELECT nationkey, name, ROW_NUMBER() AS outer FROM nation
SELECT n.outer, c.custkey, c.name, c.nationkey
WHERE c.nationkey = n.nationkey
FROM n, customer c
```

FILTER

| nationkey | name | outer | custkey | name | nationkey |
|---|---|---|---|---|---|
| ~~1~~ | ~~ARGENTINA~~ | ~~1~~ | ~~1~~ | ~~Martin~~ | ~~7~~ |
| 1 | ARGENTINA | 1 | 2 | Miguel | 1 |
| … | … | … | … | … | … |
| ~~2~~ | ~~BRAZIL~~ | ~~2~~ | ~~1~~ | ~~Martin~~ | ~~7~~ |
| ~~2~~ | ~~BRAZIL~~ | ~~2~~ | ~~2~~ | ~~Miguel~~ | ~~1~~ |
| … | … | … | … | … | … |

# Query 2 in detail

```
WITH n AS
        SELECT nationkey, name, ROW_NUMBER() AS outer FROM nation
SELECT n.outer, c.custkey, c.name, c.nationkey
WHERE c.nationkey = n.nationkey
FROM n, customer c
```

FILTER

| nationkey | name | outer | custkey | name | nationkey |
|---|---|---|---|---|---|
| 1 | ARGENTINA | 1 | 2 | Miguel | 1 |
| … | … | … | … | … | … |
| 7 | GERMANY | 7 | 1 | Martin | 7 |
| 7 | GERMANY | 7 | 3 | Tiark | 7 |
| … | … | … | … | … | … |

# Query 2 in detail

```
WITH n AS
        SELECT nationkey, name, ROW_NUMBER() AS outer FROM nation
SELECT n.outer, c.custkey, c.name, c.nationkey
WHERE c.nationkey = n.nationkey
FROM n, customer c
```

PROJECTION

| nationkey | name | outer | custkey | name | nationkey |
|---|---|---|---|---|---|
| 1 | ARGENTINA | 1 | 2 | Miguel | 1 |
| … | … | … | … | … | … |
| 7 | GERMANY | 7 | 1 | Martin | 7 |
| 7 | GERMANY | 7 | 3 | Tiark | 7 |
| … | … | … | … | … | … |

# Query 2 in detail

```
WITH n AS
        SELECT nationkey, name, ROW_NUMBER() AS outer FROM nation
SELECT n.outer, c.custkey, c.name, c.nationkey
WHERE c.nationkey = n.nationkey
FROM n, customer c
```

PROJECTION

| outer | custkey | name | nationkey |
|---|---|---|---|
| 1 | 2 | Miguel | 1 |
| … | … | … | … |
| 7 | 1 | Martin | 7 |
| 7 | 3 | Tiark | 7 |
| … | … | … | … |

# Query 2 in detail

```
WITH n AS
        SELECT nationkey, name, ROW_NUMBER() AS outer FROM nation
SELECT n.outer, c.custkey, c.name, c.nationkey
WHERE c.nationkey = n.nationkey
FROM n, customer c
```

PROJECTION

| outer | custkey | name | nationkey |
|---|---|---|---|
| 1 | 2 | Miguel | 1 |
| … | … | … | … |
| 7 | 1 | Martin | 7 |
| 7 | 3 | Tiark | 7 |
| … | … | … | … |

# Relational Results

```
SELECT nationkey, name, ROW_NUMBER() AS inner FROM nation
```

```
WITH n AS
        SELECT nationkey, name, ROW_NUMBER() AS outer FROM nation
SELECT n.outer, c.custkey, c.name, c.nationkey
WHERE c.nationkey = n.nationkey
FROM n, customer c
```

| nationkey | name | inner |
|---|---|---|
| 1 | ARGENTINA | **1** |
| 2 | BRAZIL | **2** |
| … | … | … |
| 7 | GERMANY | **7** |
| … | … | … |

| outer | custkey | name | nationkey |
|---|---|---|---|
| **1** | 2 | Miguel | 1 |
| … | … | … | … |
| **7** | 1 | Martin | 7 |
| **7** | 3 | Tiark | 7 |
| … | … | … | … |

# Relational Results -> Scala Results

| … | name | inner |
|---|---|---|
| … | ARGENTINA | 1 |
| … | BRAZIL | 2 |
| .. | … | … |
| .. | GERMANY | 7 |
| .. | … | … |

| outer | name | … |
|---|---|---|
| 1 | Miguel | … |
| … | … | … |
| 7 | Martin | … |
| 7 | Tiark | … |
| … | … | … |

```
List(
    ( Nation( …, "GERMANY" ), List(
        Customer( …, "Martin", … ),
        Customer( …, "Tiark", … ),
    ), …
)
```

# Summary

- Complete round-trip
- Nested tuples
- Avalance safety

# THANK YOU