Neidl – Privacy

Gruppe “Oben”

Anforderungen

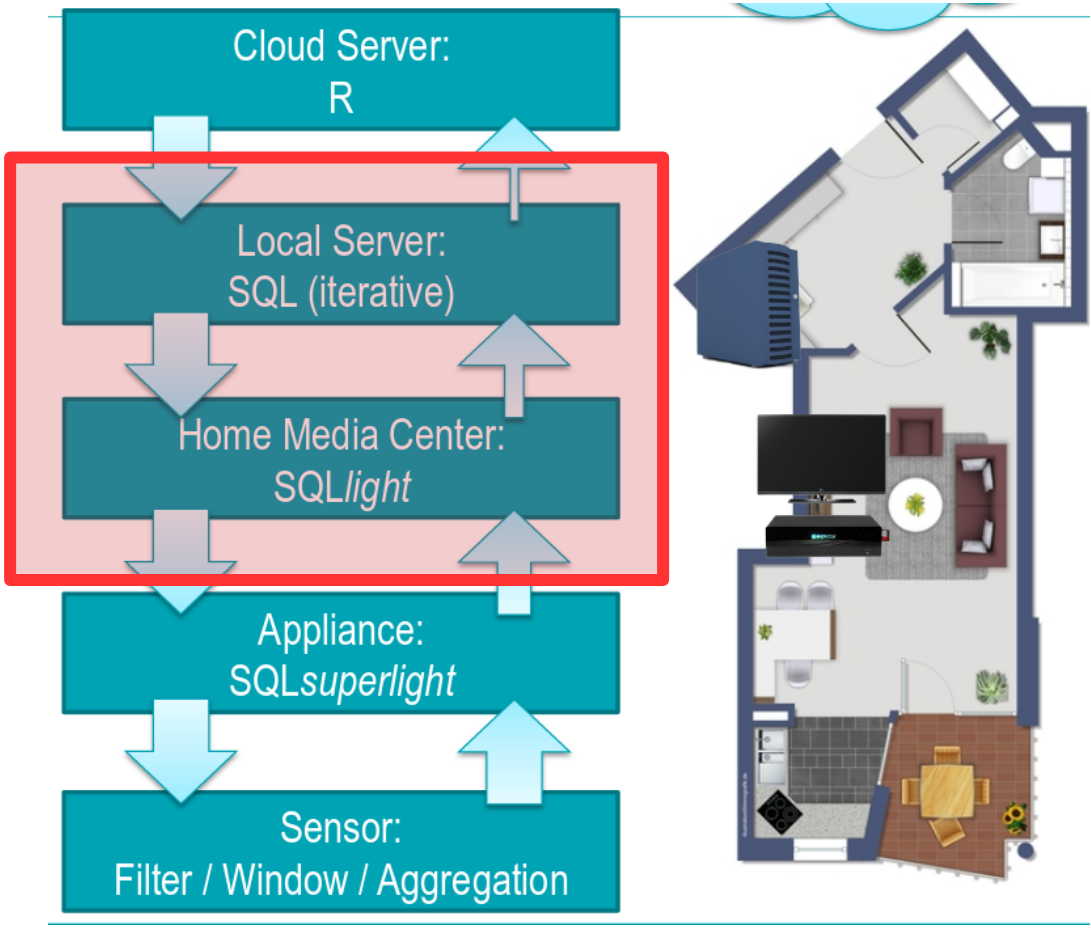
- Local Server:

  Komplexe Aggregate, OVER, PARTITION BY, etc.

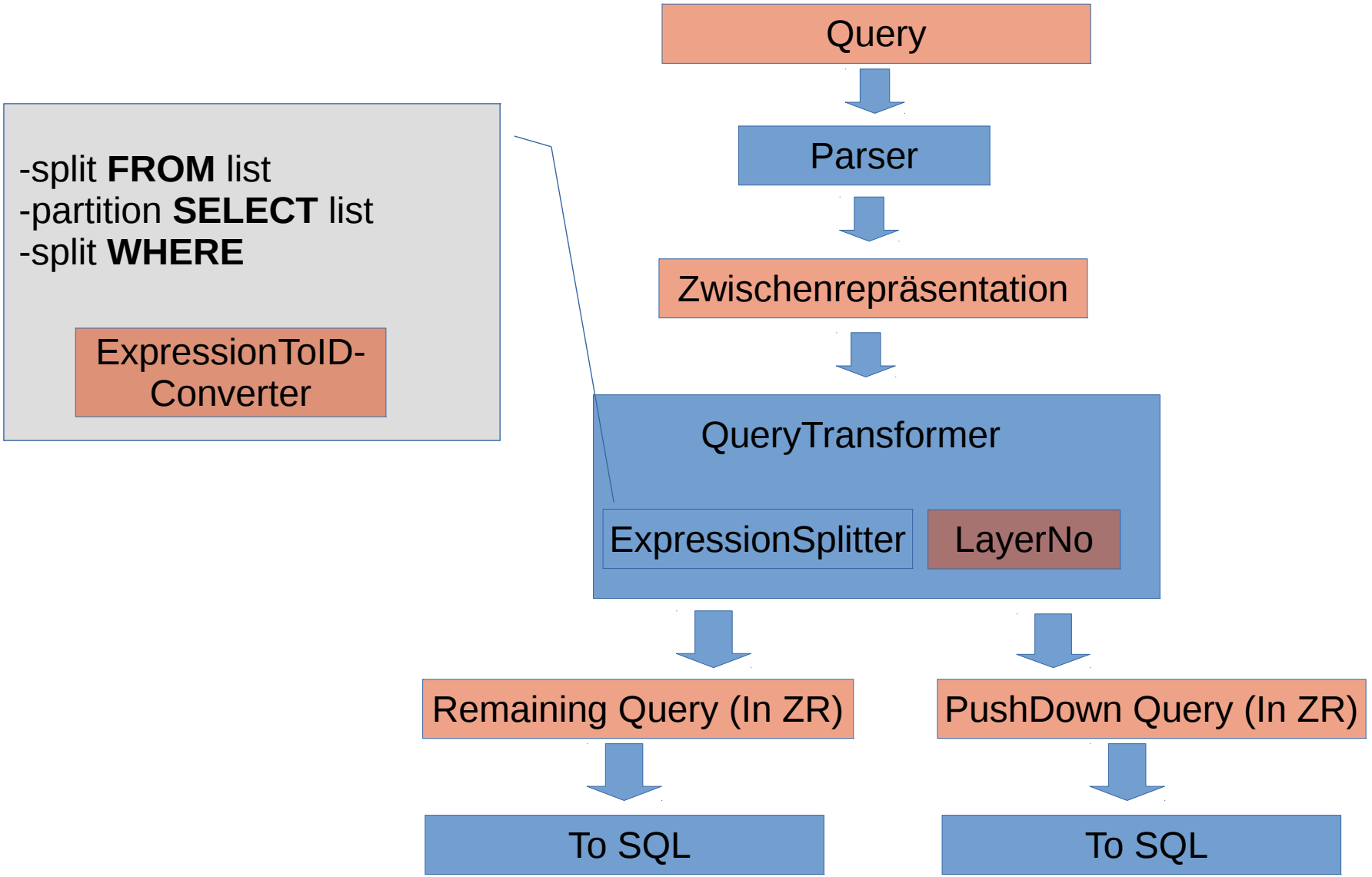
- Home Media Center:

  Einfache Aggregate, AVG, MIN, etc.

- Ziel: möglichst viel nach unten schieben
  → Privacy



Architektur Query Transformation



SQL Parser

- Übersetzt SQL Quelltext in Scala Zwischenrepräsentation
- In ANTLRv4 implementiert
- **SELECT** x,y
  FROM z
  WHERE x = y

```
simple_select_stmt returns [SimpleSelectStmt result]
 : K_SELECT v1=set_quantifier v2=result_columns
   v3=optional_from_list
   v4=optional_where_clause
   v5=optional_group_by_clause
   v6=optional_ordering_terms {$result = new
SimpleSelectStmt($v1.result, $v2.result, $v3.result,
$v4.result, $v5.result, $v6.result);}
 ;


set_quantifier returns [Option<SetQuantifier> result]
 : K_DISTINCT {$result =
Option.apply(DISTINCT$.MODULE$);}
 | K_ALL {$result = Option.apply(ALL$.MODULE$);}
 | {$result = Option.apply(null);}
 ;
```

```
SimpleSelectStatement(Nothing,
  List(ResultColumn(ColumnName(„x“),Nothing,Nothing),
       ResultColumn(ColumnName(„y“),Nothing,Nothing)),
  List(TableExpression(Table(„z“)),
  Some(BinOp(„=“,ColumnName(„x“),ColumnName(„y“))),
  Nothing,
  Nil)
```

Query Transformation

- Teile FROM Listen (NATURAL JOINS)

  Framework Call: Tabelle ist auf Layer → bleibt

  Tabelle ist nicht auf Layer → nach unten schieben

SELECT * **FROM** TAB1 **NATURAL JOIN** TAB2

remote_table ← **SELECT** * **FROM** TAB2
SELECT * **FROM** TAB1 **NATURAL JOIN** *remote_table*

Query Transformation

- Teile Projektionsliste
  split(e:Expr) => (Expr,List[ResultColumn])
- Extrahiere so viele Sub-Expressions wie möglich
- Ersetzen von Sub-Expressions durch ColumnNames (Hash-Wert von Sub-Expression)
- Außerdem referentielle Layer Transparenz
  → Auf ColumnNames muss projiziert werden, falls nicht auf aktuellem Layer vorhanden und Expression nicht nach unten geschoben werden darf.
- Nur FunctionCalls, die auf unteren Layern unterstützt werden, können nach unten geschoben werden.
- Aggregat vor OVER Klausel muss auf Local Server Layer verbleiben

Query Transformation

- **WHERE** wird als Expression behandelt → analog zu Projektionsliste
- Spezielle Behandlung für Konjunktionen

  - Teil-Terme extrahieren und nach unten schieben wenn möglich

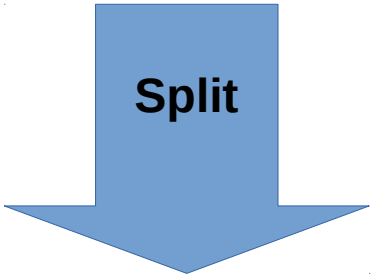
  - Abspaltung von Klauseln die nach unten geschoben werden können

  →Frühere Filterung von Zeilen → Privacy 

Query Transformation

- **Beispiel Sensor**

```
SELECT MIN(sensor.time),
    AVG(sensor.x + sensor.y) AS avg_sy,
    AVG(sensor.x + sensor.y) OVER(PARTITION BY sensor.key)
FROM sensor
WHERE ((sensor.i + sensor.j) < 20) AND
   (sensor.time BETWEEN 1 AND 10)
```



Query Transformation

- Beispiel Sensor: Split auf Local Server Ebene

```
remote_table ← SELECT (MIN(sensor.time) )  AS
id_56e75d9e,
      (AVG((sensor.y + sensor.x)) )  AS id_b6e62d57,
      sensor.x  AS id_d2404843,
      sensor.y  AS id_ea386849,
      sensor.key  AS id_6b4c67f9
FROM sensor
WHERE (((sensor.i + sensor.j) < 20) AND
     (sensor.time  BETWEEN 1 AND 10))

SELECT id_56e75d9e, id_b6e62d57 AS avg_sy,
  (AVG((id_d2404843 + id_ea386849)))
    OVER (PARTITION BY id_6b4c67f9)
FROM remote_table
```

Query Transformation

- Beispiel Sensor: Split auf HMC Ebene

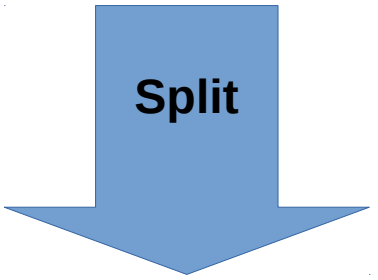
remote_table ← **SELECT** (sensor.y + sensor.x) **AS** id_acdbc907,
    sensor.time **AS** id_5d283dd0,
    sensor.x **AS** id_d2404843,
    sensor.y **AS** id_ea386849,
    sensor.key **AS** id_6b4c67f9
FROM sensor
WHERE (((sensor.i + sensor.j) < 20) **AND**
    (sensor.time **BETWEEN** 1 **AND** 10))

SELECT (**MIN**(id_5d283dd0)) **AS** id_56e75d9e,
    (**AVG**(id_acdbc907)) **AS** id_b6e62d57,
    id_d2404843 **AS** id_d2404843,
    id_ea386849 **AS** id_ea386849,
    id_6b4c67f9 **AS** id_6b4c67f9
FROM *remote_table*

Query Transformation

- **Beispiel Multi-Layer**

```
SELECT sensor.y + analytics.geo_location AS
sy_plus_geo,
cloud_meta.aws_user * (sensor.x + sensor.y) AS
magic_hash
FROM sensor NATURAL JOIN analytics
    NATURAL JOIN cloud_meta
WHERE cloud_meta.aws_acl = 'admin' OR
analytics.time_zone = 'UTC+1'
```



Query Transformation

- Beispiel Multi-Layer: LocalServer Split

```
remote_table ← SELECT
(sensor.y + analytics.geo_location)  AS id_254f6e78,
 (sensor.x + sensor.y)  AS id_f209090c,
(analytics.time_zone = 'UTC+1')  AS id_01aeb1ae
FROM (sensor NATURAL JOIN analytics)
```

```
SELECT id_254f6e78 AS sy_plus_geo,
    (cloud_meta.aws_user * id_f209090c)  AS
    magic_hash
FROM (cloud_meta  NATURAL JOIN remote_table)
WHERE ((cloud_meta.aws_acl = 'admin')
    OR id_01aeb1ae)
```

Query Transformation

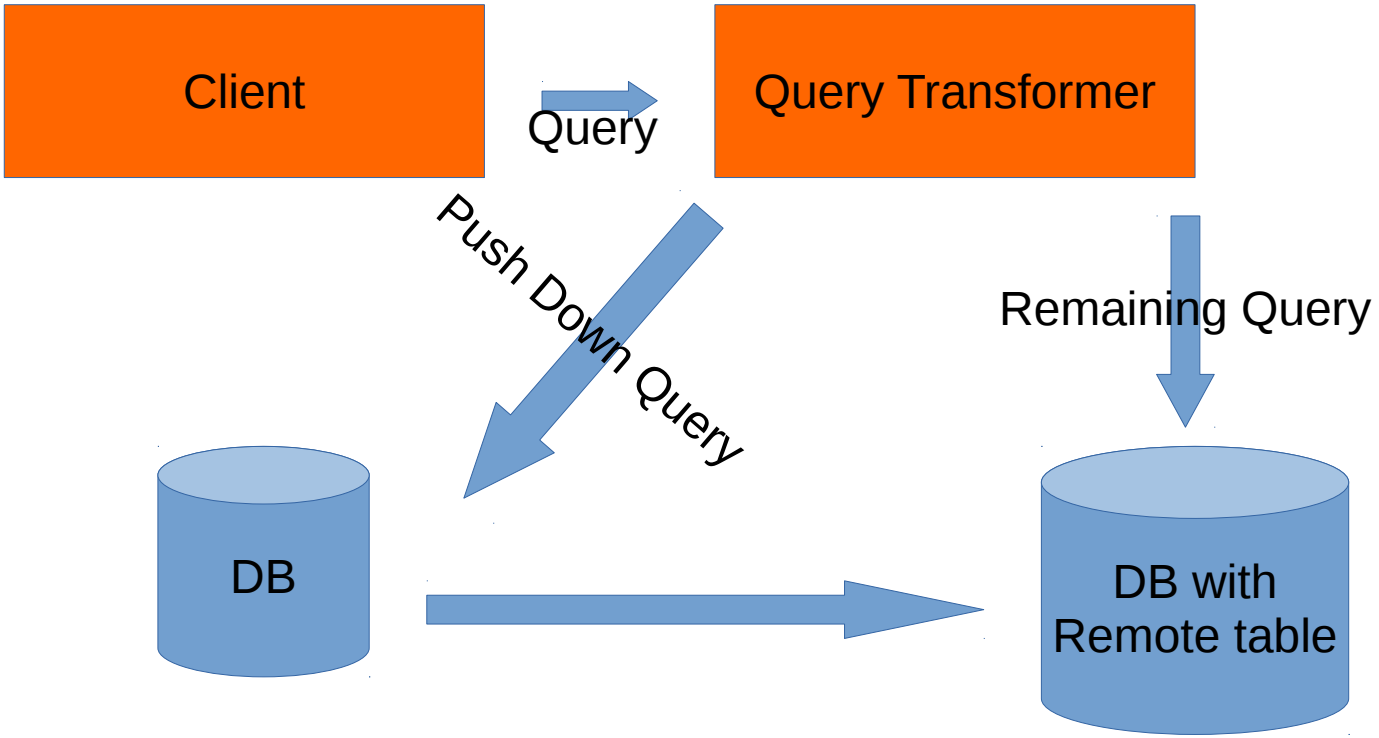
- Beispiel Multi-Layer: HMC Split

```
remote_table ← SELECT
    (sensor.x + sensor.y)  AS id_f209090c,
    sensor.y  AS id_ea386849
FROM sensor
```

```
SELECT
    (id_ea386849 + analytics.geo_location)  AS id_254f6e78,
    id_f209090c AS id_f209090c,
    (analytics.time_zone = 'UTC+1')  AS id_01aeb1ae
FROM (analytics  NATURAL JOIN remote_table)
```

Datenbank Anbindung (Konzept)

- Derzeitig nur für PostgresSQL



Datenbank Anbindung (Implementierung)

- Auf Datenbankseite: Extension pljava (erlaubt Implementierung)
- Copy Remote Table Funktion anlegen, Parameter: Datenbankverbindung, Query, Tabellenname.
- Führt Query aus.
- Iteriert über das ResultSet und Column-Handler-Chain.
- Legt Ergebnistabelle an.

Demo

Ausblick (Offene Probleme)

- Kombination aus INNER JOIN und NATURAL JOIN nicht kommutativ
- GROUP BY nicht trivial transformierbar (abhängig von Aggregat)
- ORDER BY ändert Semantik wenn Reihenfolge der Ordering Terms verändert wird
- Set Quantifier
- Aufspalten von Projektionsliste in Aggregat- und nicht Aggregat-Teil
- Substituierte Namen dürfen bestimmte Länge nicht überschreiten → Hash
- Umbenennung von Tabelle mit Alias funktioniert noch nicht.
- Anbindung an PArADISE noch nicht vollständig
- Berücksichtigt noch keine Nutzer privacy policies

Neidl – Privacy

Gruppe “Oben”