Querify Labs

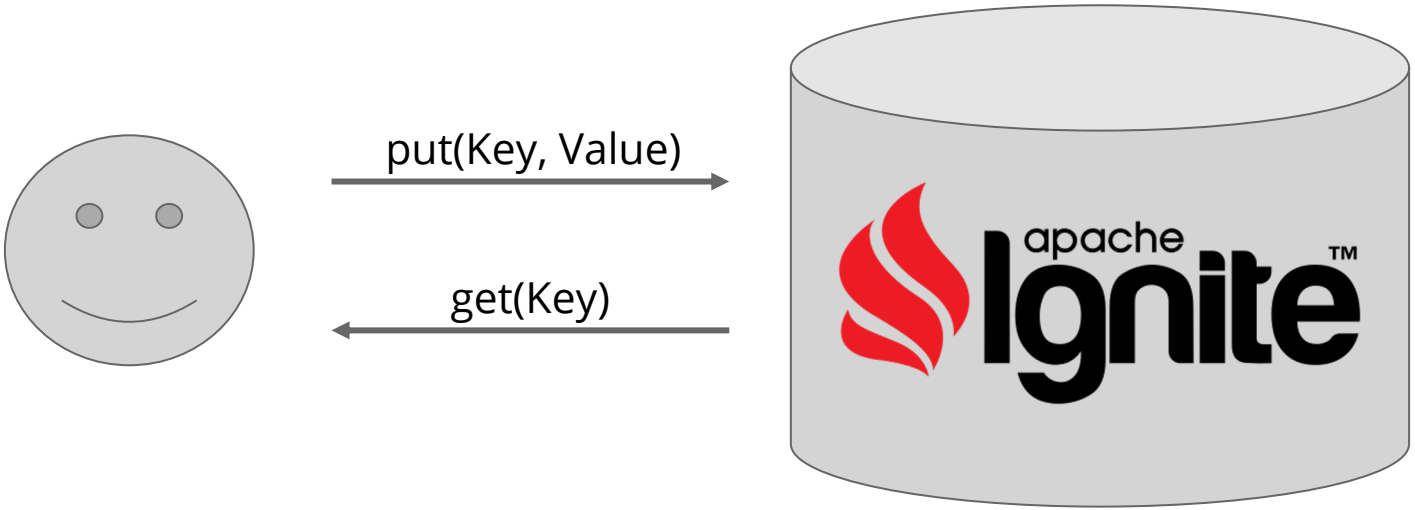
# Как прикрутить SQL к чему угодно при помощи Apache Calcite

Роман Кондаков
Querify Labs

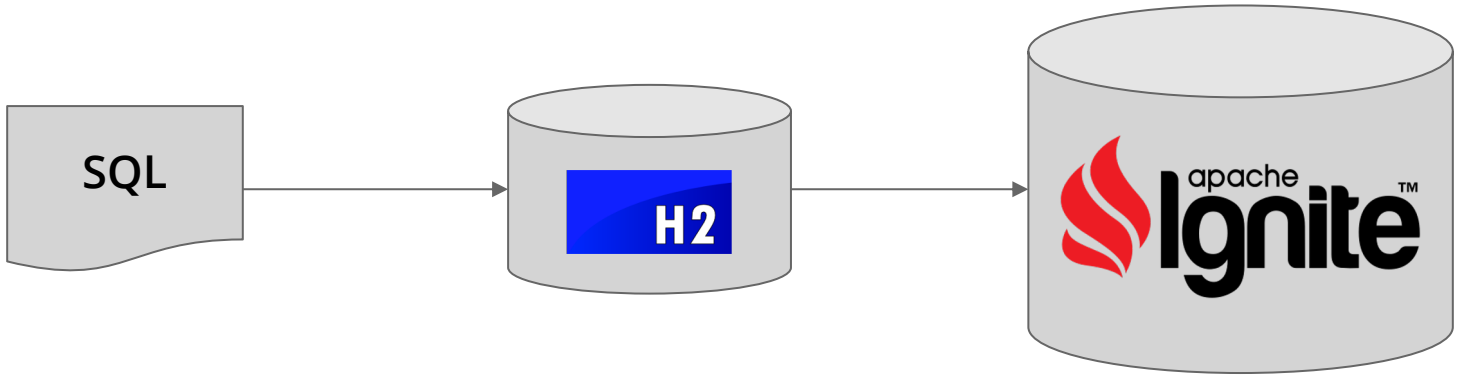
# Обо мне

- Работаю в Querify Labs
  - Консультируем по созданию СУБД и использованию Apache Calcite
- Ex-Yandex, недолго занимался Yandex Query Language
- Ex-Gridgain, делал SQL движок для Apache Ignite

# SQL в Apache Ignite. Попытка №1



# SQL в Apache Ignite. Попытка №1



# SQL в Apache Ignite. Попытка №1

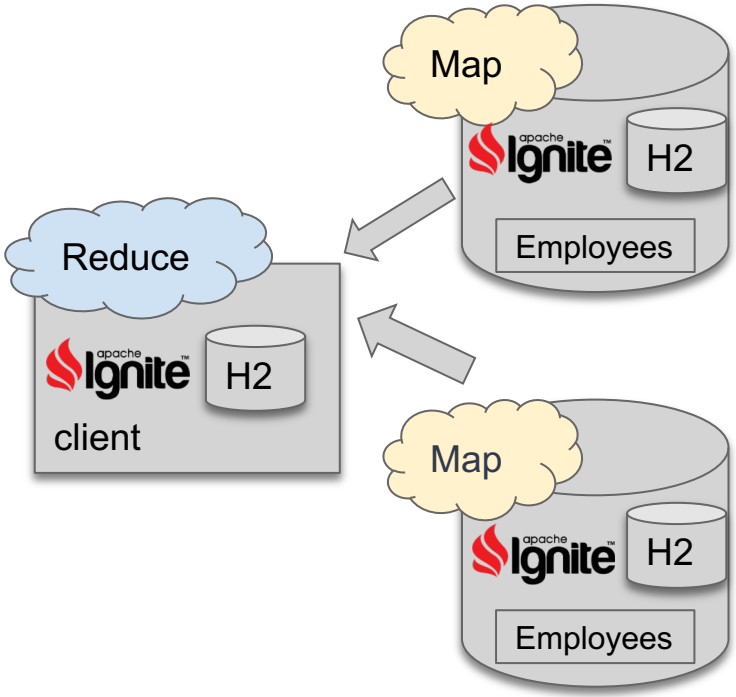
```
SELECT AVG(salary) FROM employees
```

=

```
Map query:
SELECT SUM(salary) as sum0,
COUNT(salary) as count0
FROM employees
```

+

```
Reduce query:
SELECT SUM(sum0) / SUM(count0)
FROM resultTable
```



# SQL в Apache Ignite. Попытка №1

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

2 x Map + 2 x Reduce == Not OK

1 x Map + 1 x Reduce == OK

=

```
X = SELECT AVG(emps.salary) FROM emps
```

= Map qry + Reduce qry

+

```
SELECT * FROM emps WHERE emps.salary =
X
```

= Map qry + Reduce qry

# Как починить SQL в Apache Ignite?

Варианты:

- Усовершенствовать старый движок (мрачные перспективы)
- Написать с нуля (долго и рискованно)
- Поискать готовые решения (а вот это интересно)

# Какое-такое готовое решение?

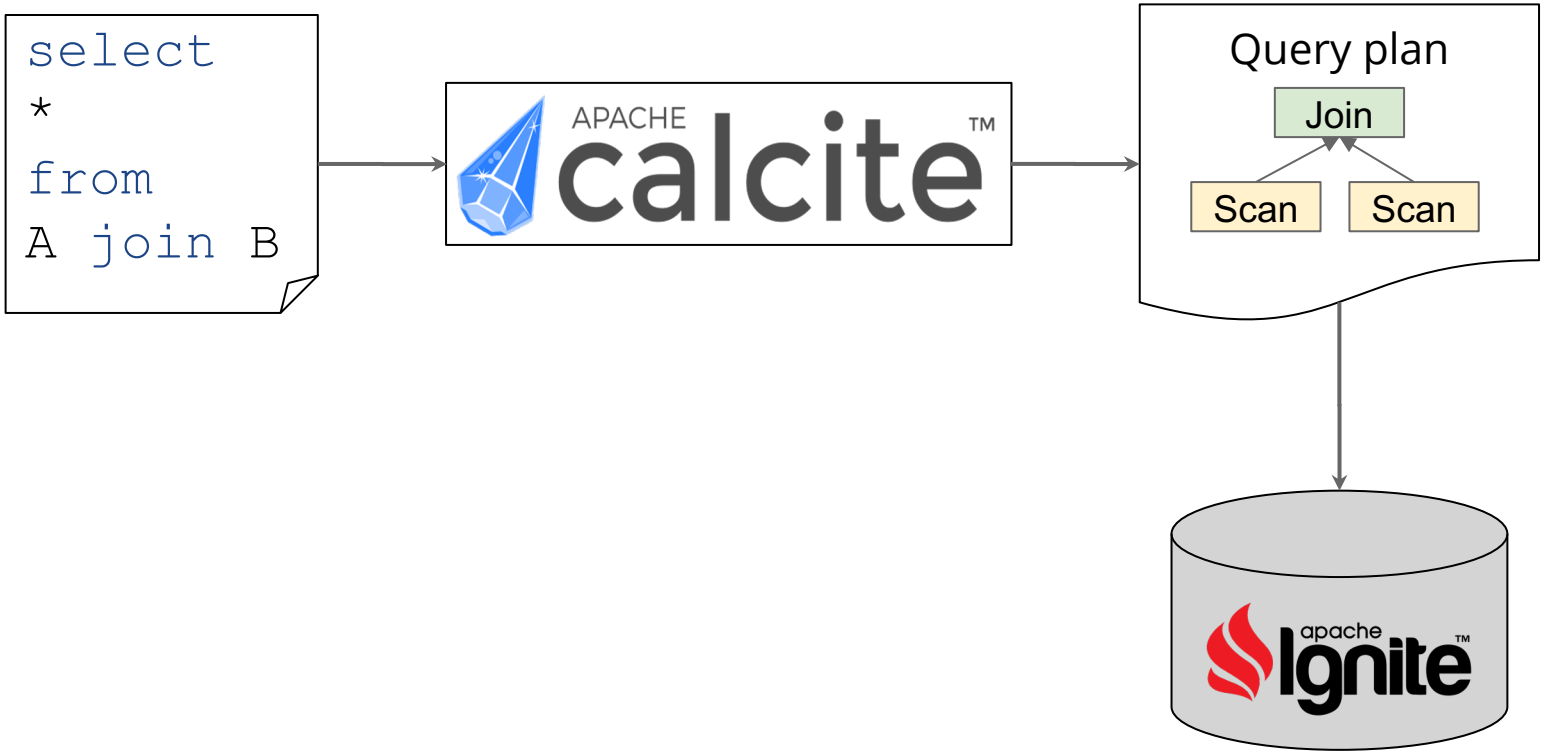
Apache Calcite - фреймворк для создания SQL БД


- Как мы при помощи Calcite чиним SQL в Apache Ignite
- Внутренности Apache Calcite
- Нюансы для распределенных систем

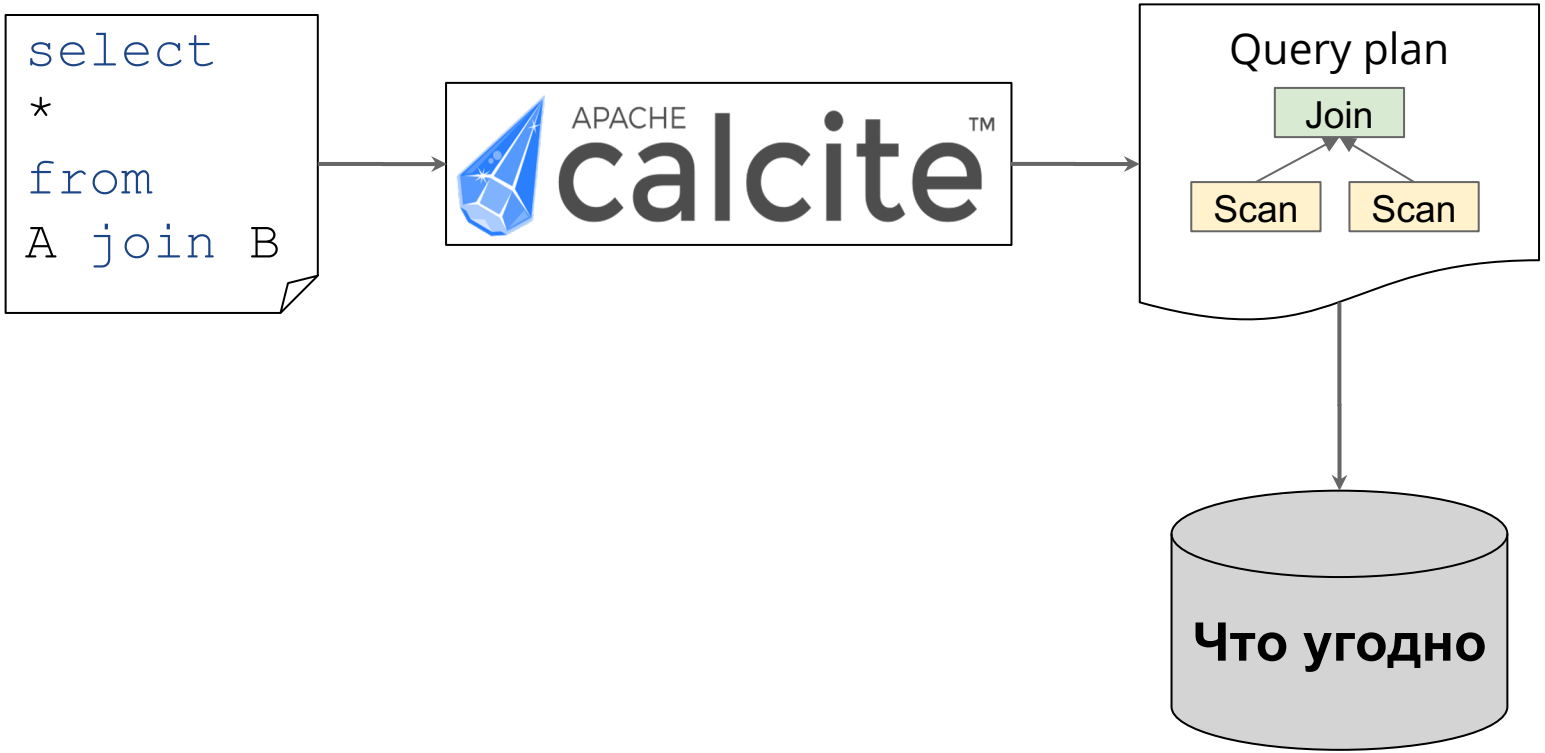
# Кто еще использует?

- Стартовал в начале 2000-х
- Apache с 2013г.

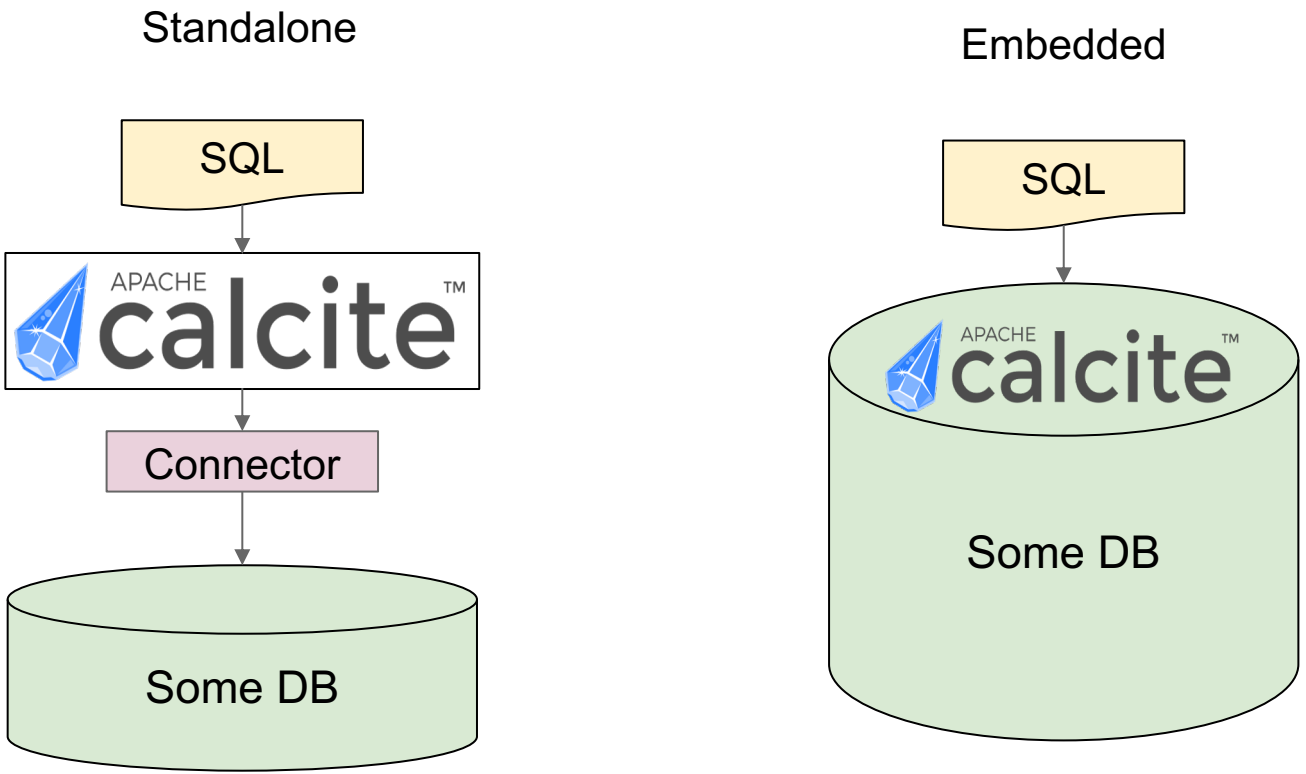

# А как использовать Кальцит?



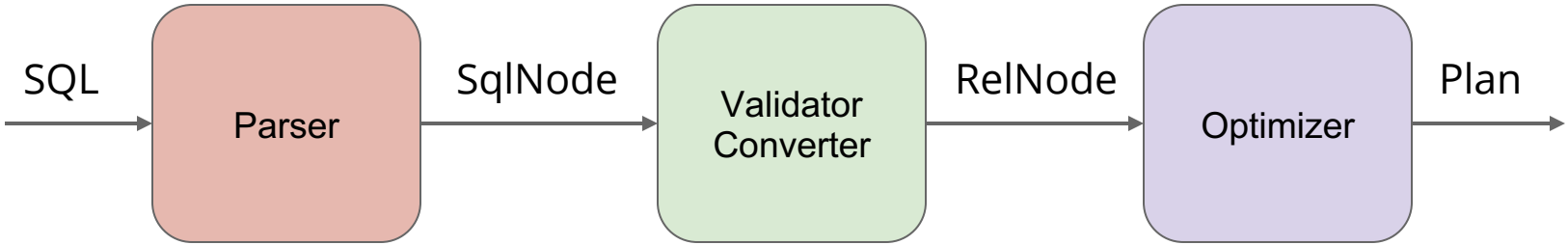
# А как использовать Кальцит?



# Варианты интеграции Apache Calcite

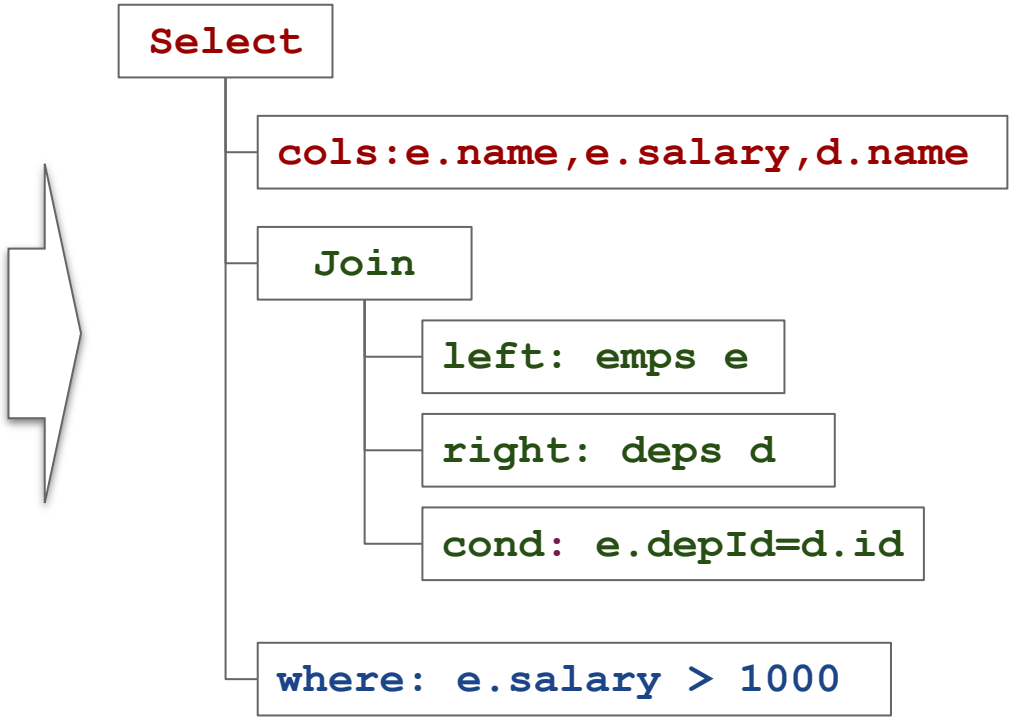


# Основные компоненты Кальцита

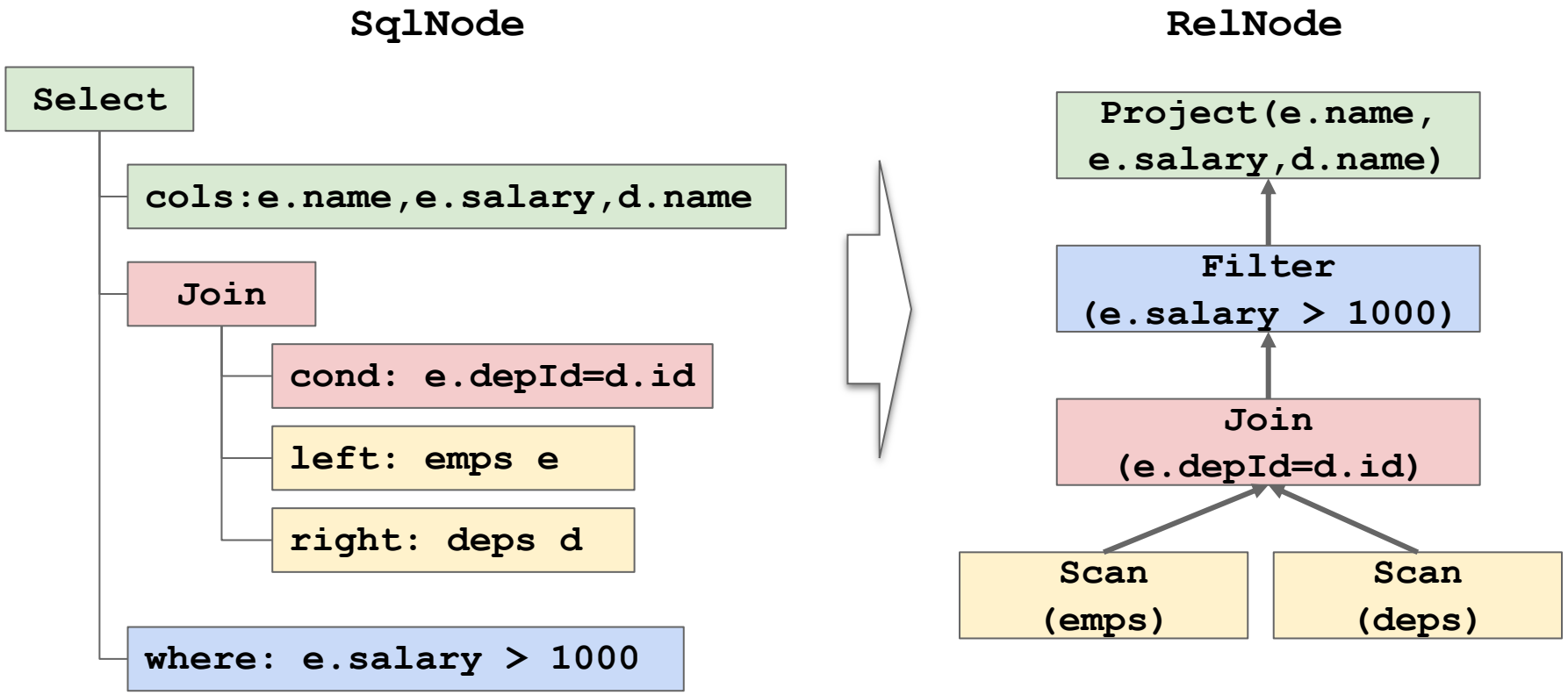


# Парсер

```
SELECT
  e.name, e.salary, d.name
FROM
  emps e JOIN deps e
ON
  e.depId = d.id
WHERE
  e.salary > 1000
```

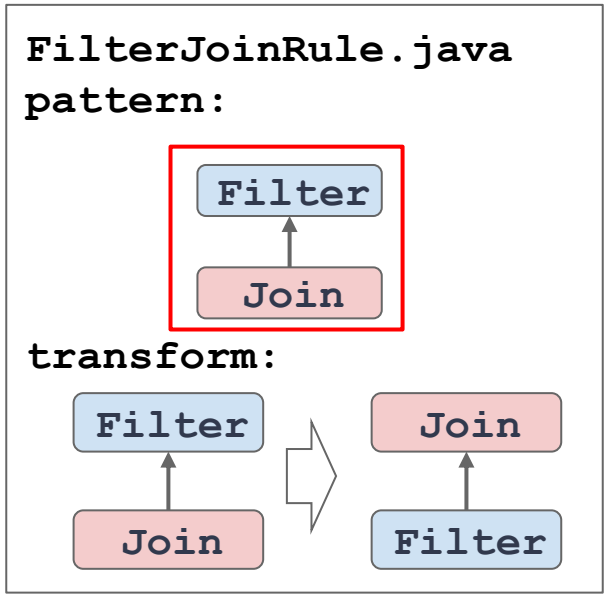


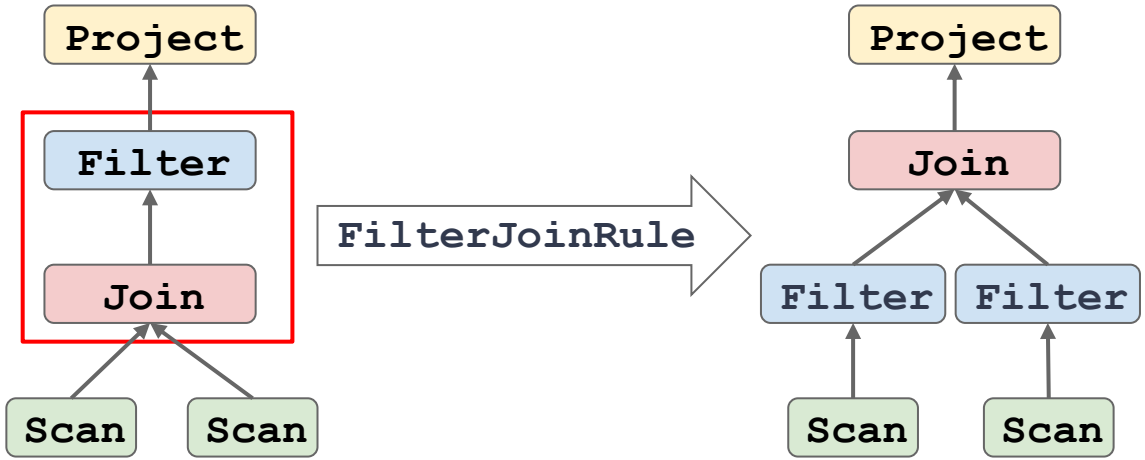
# Конвертер


SqlNode
RelNode
Select
cols:e.name,e.salary,d.name
Join
cond: e.depId=d.id
left: emps e
right: deps d
where: e.salary > 1000
Project(e.name, e.salary,d.name)
Filter (e.salary > 1000)
Join (e.depId=d.id)
Scan (emps)
Scan (deps)

# Оптимизатор и его правила

- Конфигурируется при помощи правил



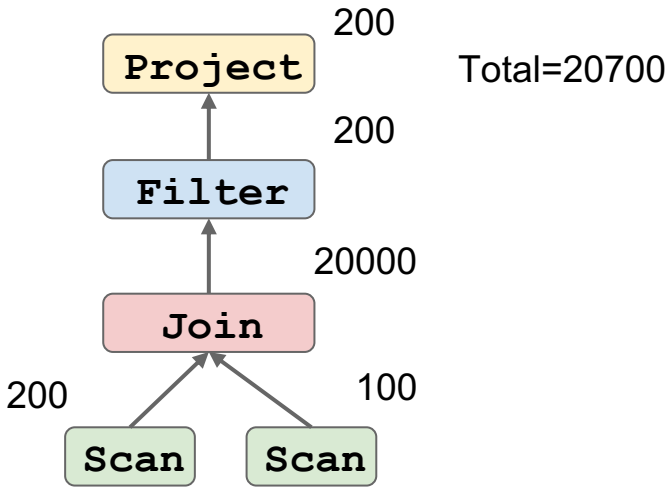
# HepPlanner - эвристический (rule-based)  оптимайзер

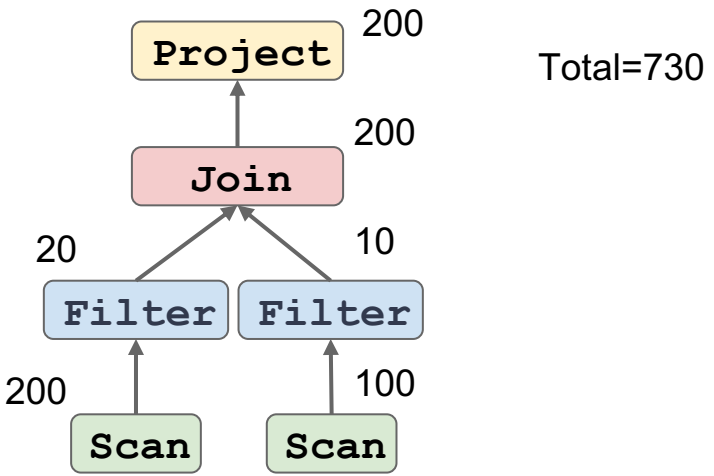
- Применяет правила пока может (возможно зацикливание)
- Быстрый
- Используется для всегда оптимальных преобразований

```
while(canApplyRule()) {
        applyRule();
}
```

# VolcanoPlanner - cost based optimizer

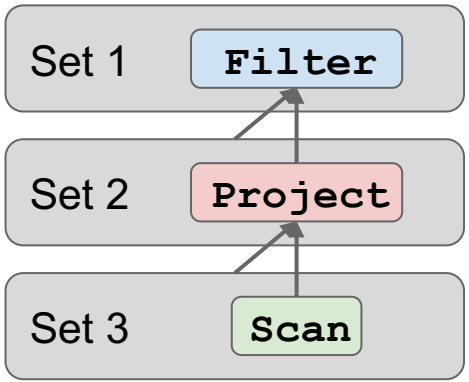
- Учитывает кост операторов
- Хранит все модификации дерева запроса в структуре MEMO
- Сначала применяет правила, потом ищет в MEMO лучший план



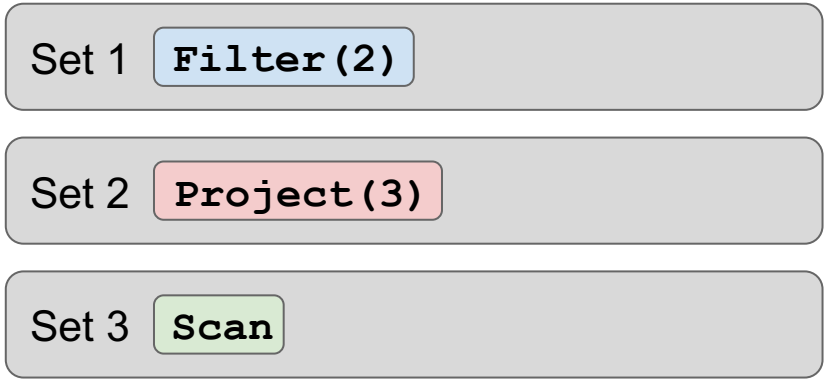



# MEMO

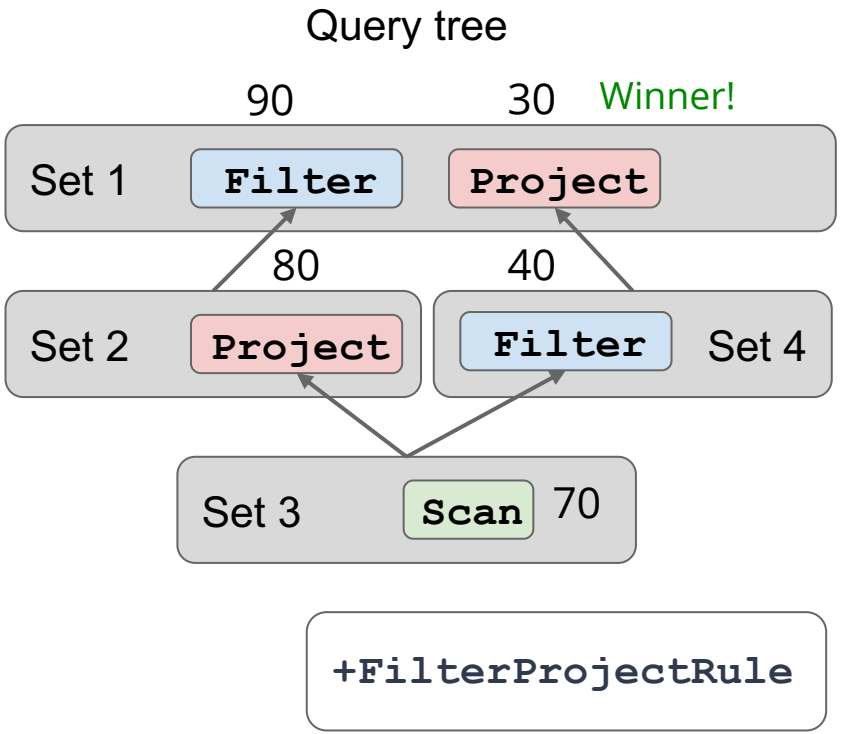
Query tree

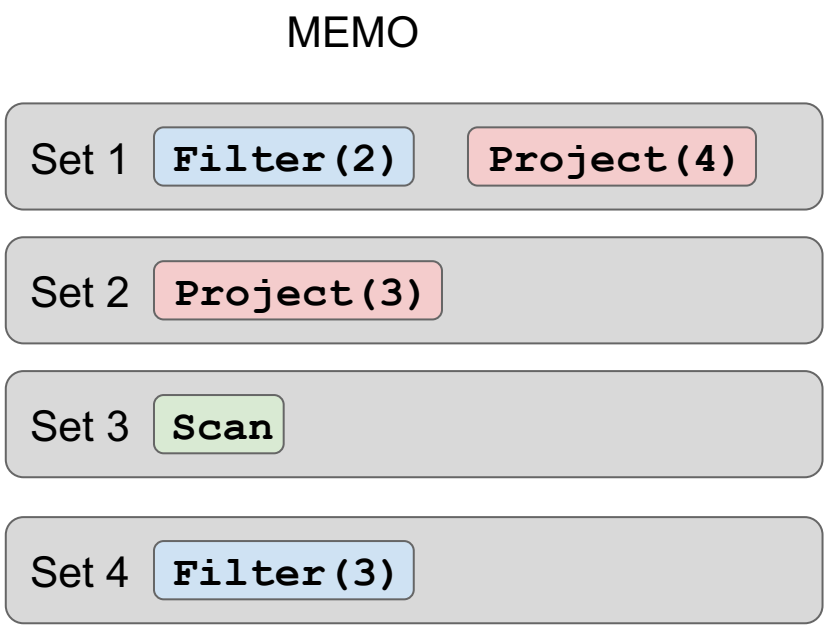



MEMO

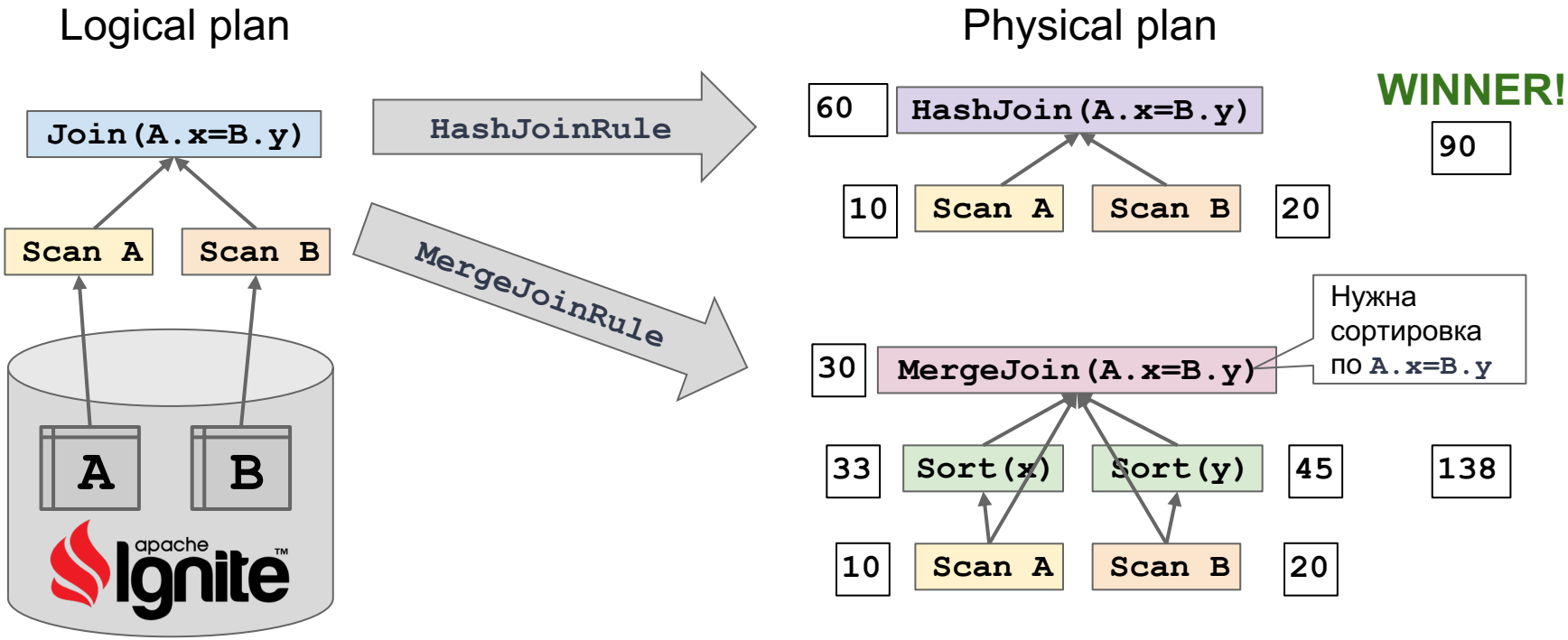


# MEMO

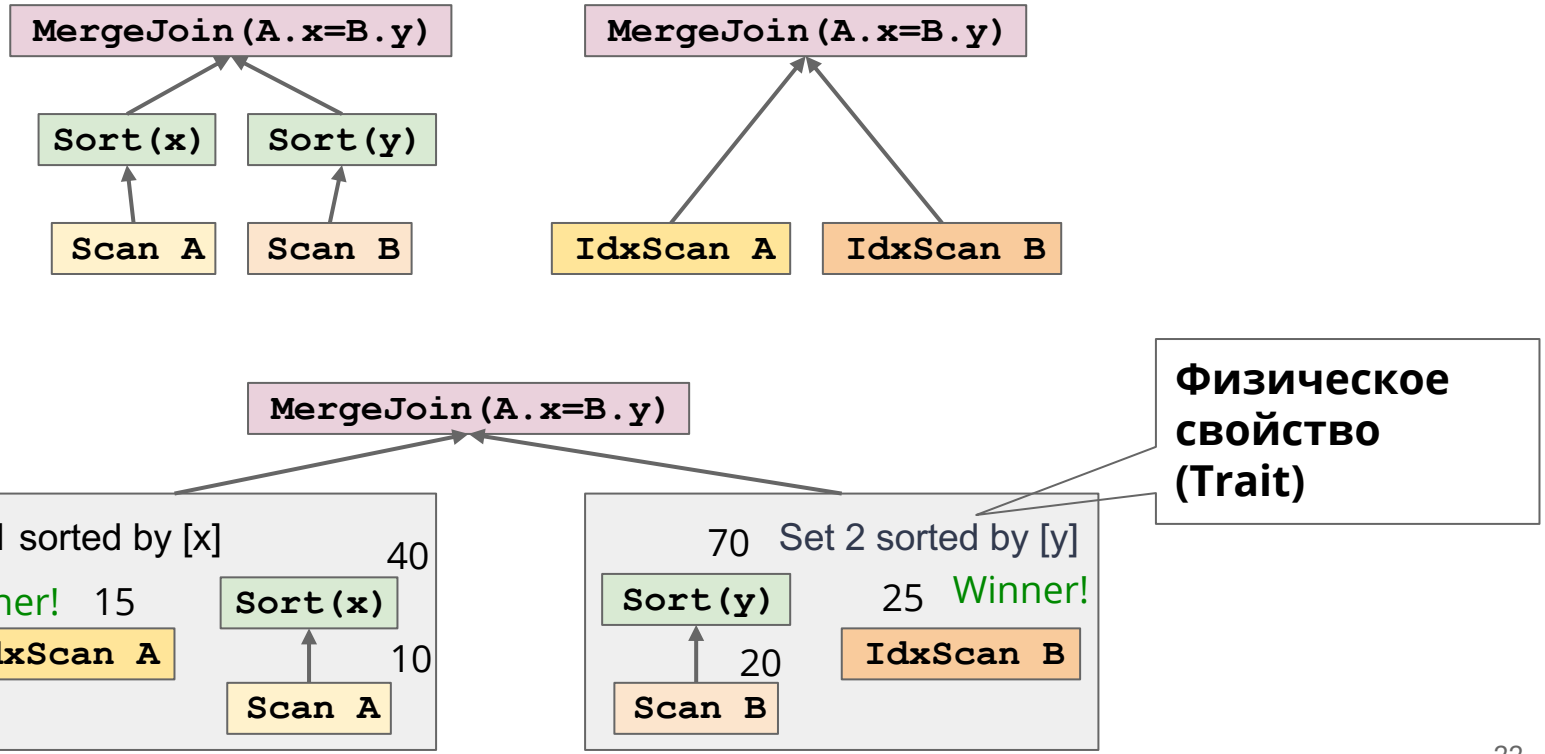

Query tree
90
30
Winner!
Set 1
Filter
Project
80
40
Set 2
Project
Filter
Set 4
Set 3
Scan
70
+FilterProjectRule


MEMO
Set 1
Filter(2)
Project(4)
Set 2
Project(3)
Set 3
Scan
Set 4
Filter(3)

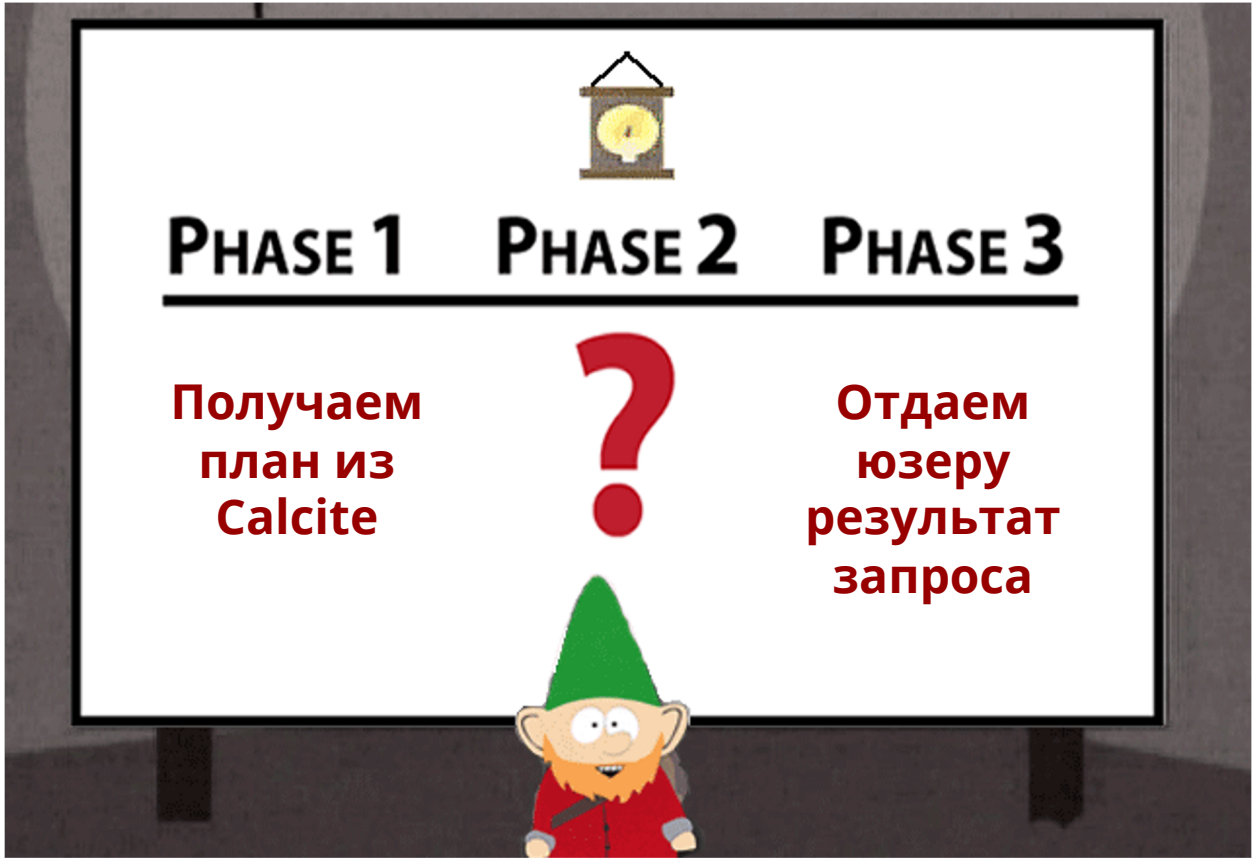
# VolcanoPlanner - физическое планирование



# Сортировка и причем тут трейты

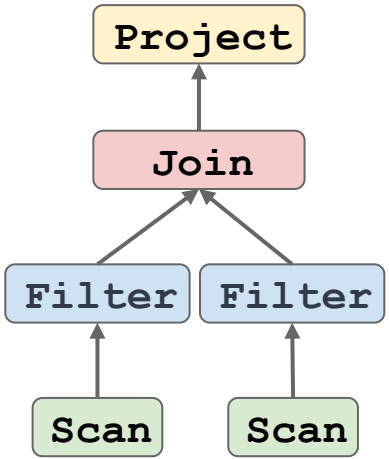


# Что делать с планом?

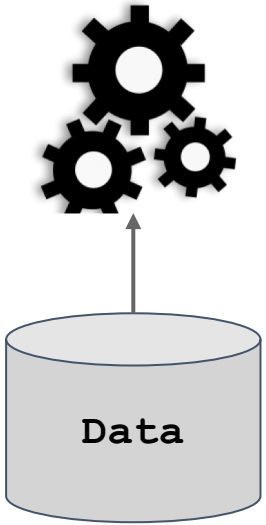


# Рантайм?

**Query plan**




**Runtime**

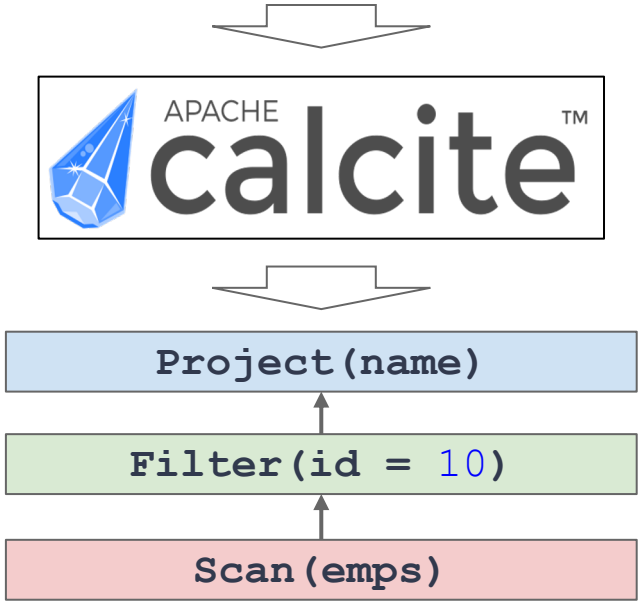



**Query result**

| | |
|---|---|
| 1 | a |
| 2 | b |
| 3 | c |
| 4 | d |
| 5 | e |

# И как сделать рантайм?

```
SELECT name FROM emps WHERE id = 10
```



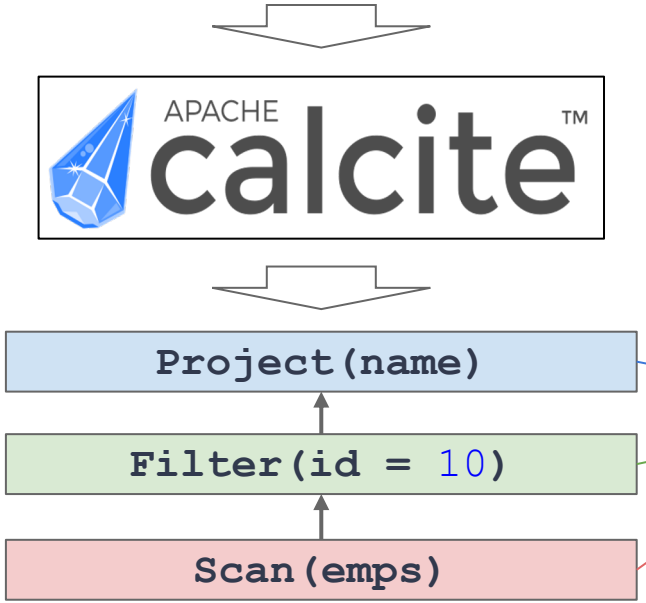

```
for row : child.next()
 emit(row[name]))
```

```
for row : child.next()
  if row[id] = 10
    emit(row)
```

```
for row : emps
  emit(row)
```

# Кодогенерация

```
SELECT name FROM emps WHERE id = 10
```



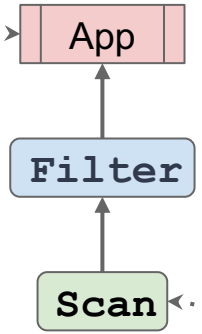

```
public class Foo {
  private Scan scan;

  public Row next() {
    while (scan.hasNext()) {
      Row row = scan.next()
      if (row.id = 10)
        return new Row(row.name)
    }
  }
}
```

# Еще рантаймы

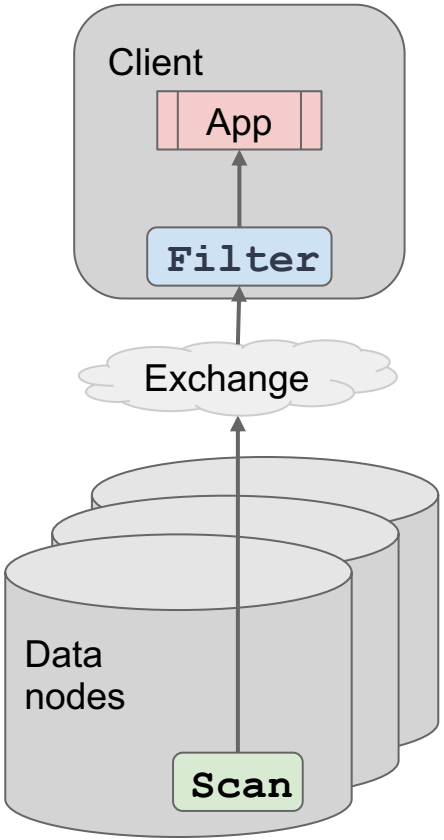
- Векторизация (SIMD)
- Векторизация + кодогенерация
- GPU-acceleration (Kinetica, SQream, BlazingDB)
- FPGA-acceleration (Netezza)

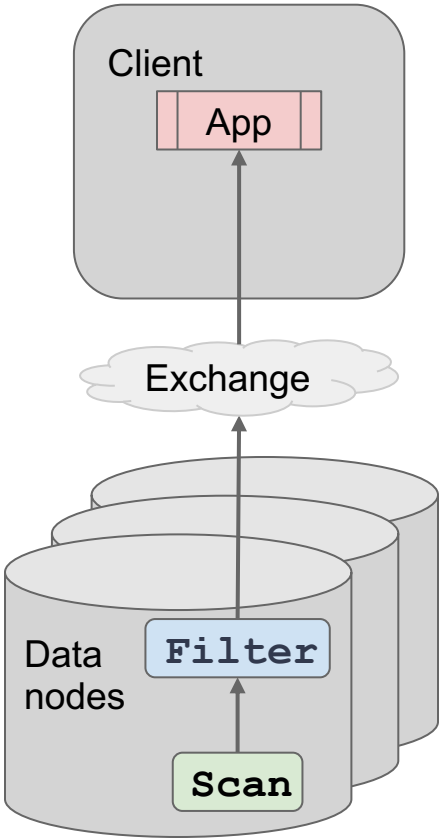
# А как быть в распределенной системе?

Исполняется на клиентском узле



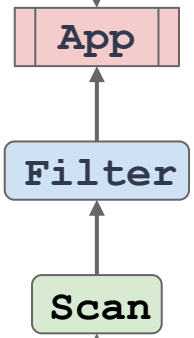

Исполняется где-то в кластере



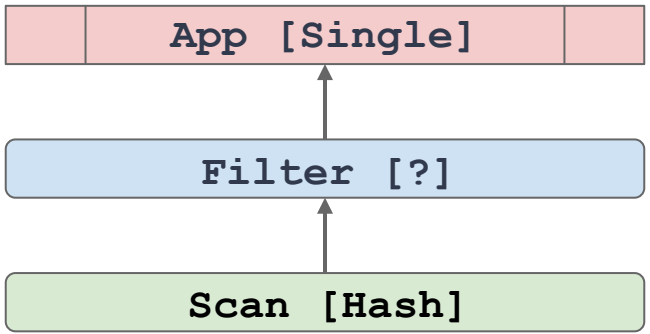
# Как сказать кальциту о распределенных данных?

Исполняется на клиентском узле



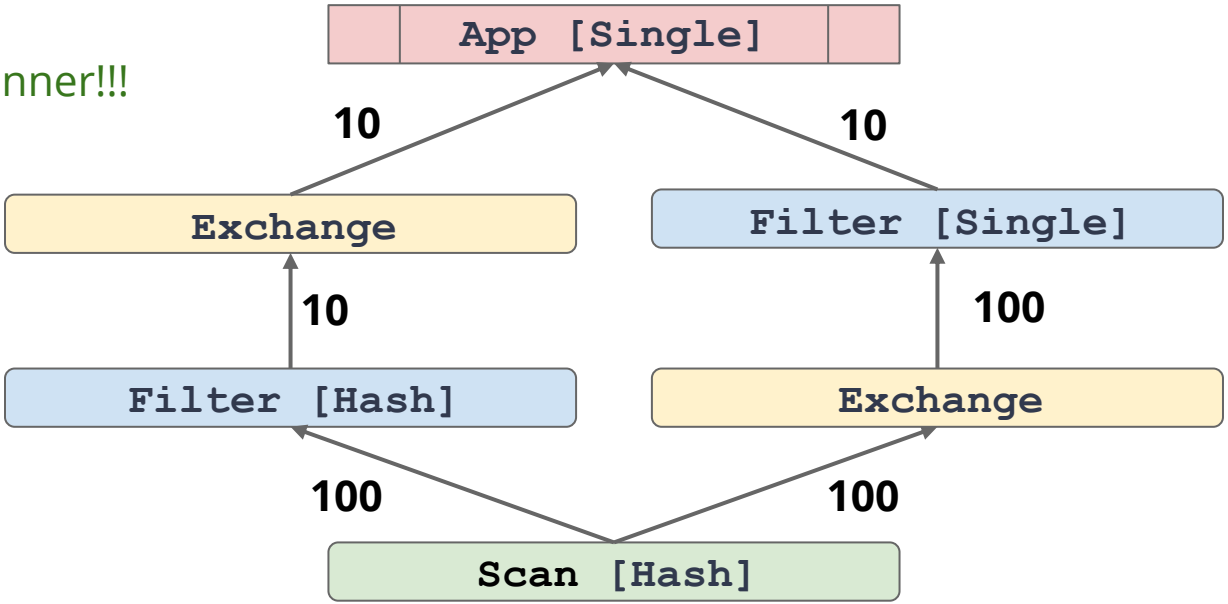

Исполняется где-то в кластере

Трейт распределения в кластере
(как сортировка, только распределение)



# Оптимизатор и трейты

Winner!!!



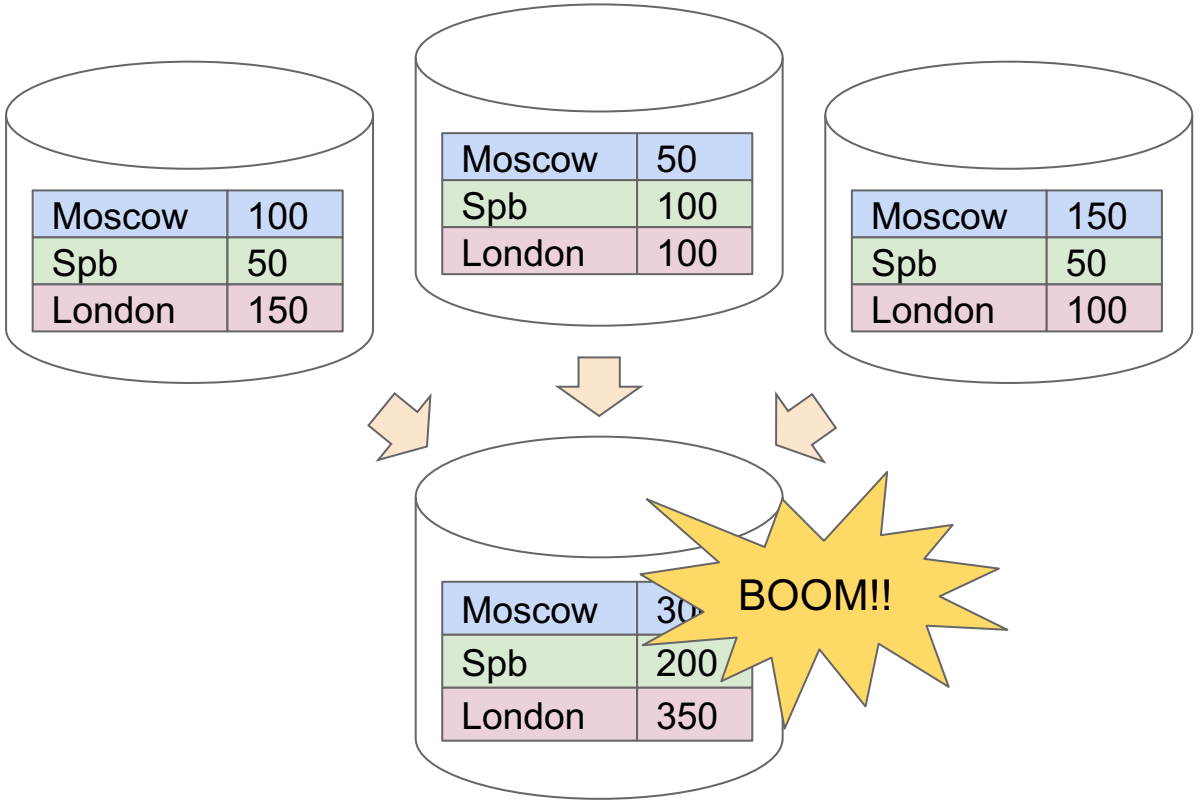
# Apache Ignite смог!

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```


```
SingletonExchange // Отправляем полученный результат на клиента
  HashJoin(condition=[salary=#0]) // Джойним emps со средним
    Scan(table=[emps]) // Снова сканируем emps на всех нодах
    BroadcastExchange // Отправляем полученное среднее на все ноды
      Project(#0=[#0 / #1]) // Получаем среднее число как сумма/количество
        Agg(#0=[SUM(#0)], #1=[SUM(#1)]) // Считаем глобальные агрегаты
          SingletonExchange // Отправляем локальные агрегаты на клиента
            Agg(#0=[SUM(salary)], #1=[COUNT(1)]) // Считаем локальные агрегаты
              Scan(table=[emps]) // Сканируем emps на всех нодах
```

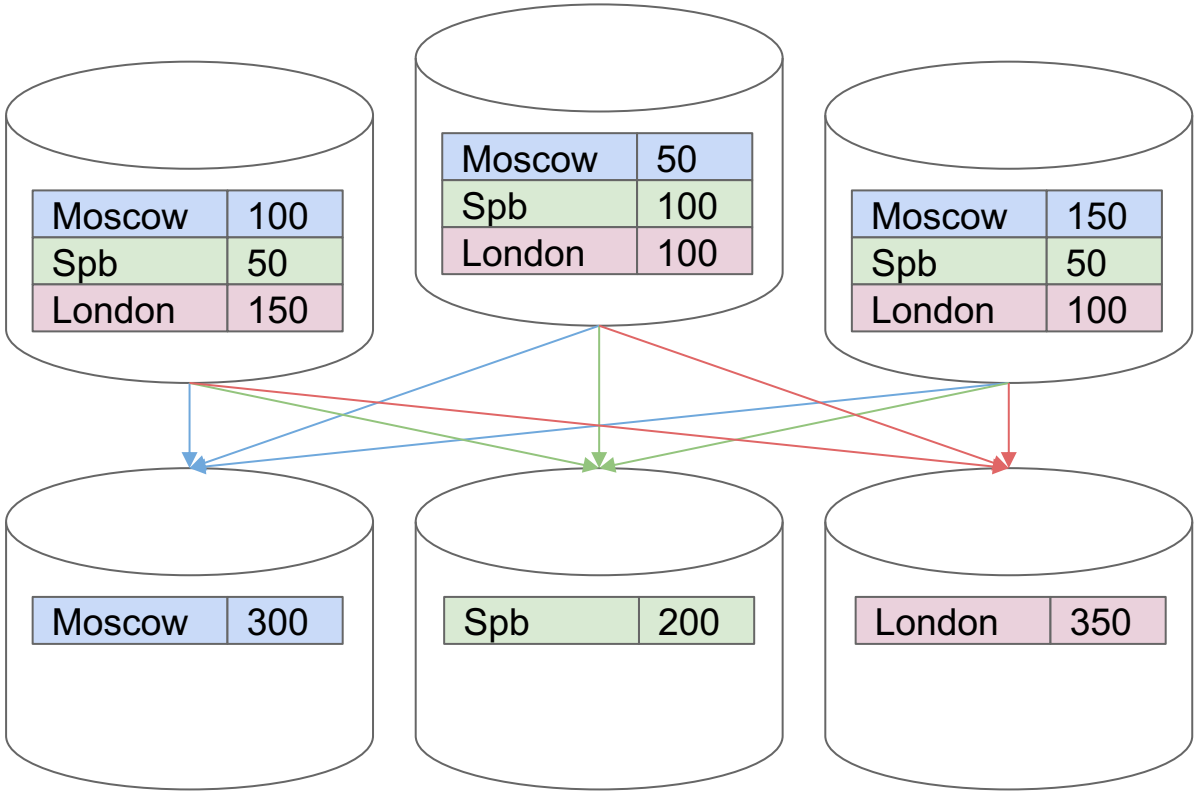
# Не желательно отправлять все данные на один узел

```
SELECT
        city,
        SUM(amount)
FROM sales
GROUP BY city
```



# Не желательно отправлять все данные на один узел

```
SELECT
        city,
        SUM(amount)
FROM sales
GROUP BY city
```



# А можно кастомизировать Calcite?

- Добавить свои операторы RelNode
- Добавить свои правила оптимизатора
- Переопределить косты у своих и встроенных операторов
- и многое другое

# Полезные ссылки

- Сайт проекта https://calcite.apache.org/
- Хорошая презентация с примерами кода: https://www.slideshare.net/JordanHalterman/introduction-to-apache-calcite
- Полезная статья про Calcite на arxiv.org: https://arxiv.org/pdf/1802.10233.pdf
- Volcano/Cascades optimizer papers
  https://www.cse.iitb.ac.in/infolab/Data/Courses/CS632/Papers/Cascades-graefe.pdf
  https://users.cs.fiu.edu/~fortega/storage/cop5725/Topic%20List%20Papers/15-optimizer2/IDEAS01.pdf

# Вопросы?

e-mail: rkondakov@querifylabs.com