Python和大数据计算平台的结合

秦续业

2016.9

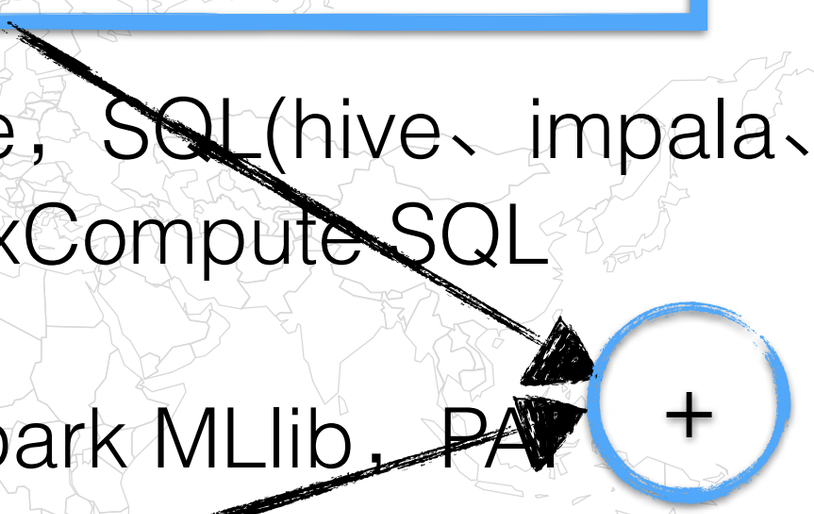
关于我

- 阿里云大数据事业部
- 微博：@秦续业
- GitHub：@chineking


Semi-Structured
Unstructured data
Volume
Useful
Metadata
Petabytes
analysis
content
BIG DATA
People driven
Decision making
Data analysis
Unstructured content
Zettabyte
Framework
Concept extraction
Smart content
Text analytics
database
SAS
Semantic Metadata

数据分析和机器学习

- Hadoop Ecosystem、MaxCompute(big data)
  - 数据分析：MapReduce，SQL(hive、impala、drill)，spark SQL，MaxCompute SQL
  - 机器学习：mahout、spark MLlib，PAI
- PyData(small data)
  - 数据分析：numpy，pandas，scipy，dask，blaze
  - 机器学习：scikit-learn

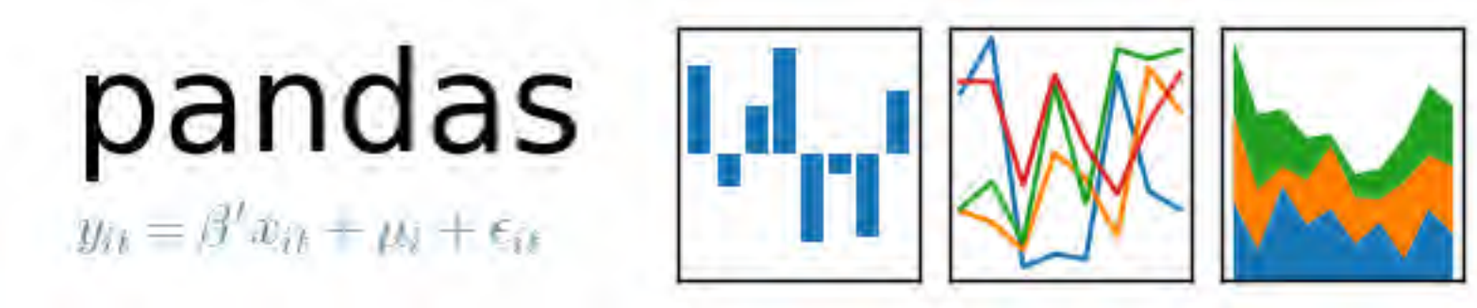


MaxCompute

- TB/PB、多租户、开箱即用、安全
- SQL
  - 大数据领域主要的分析工具
    - 简单，容易上手，描述型
- Tunnel数据上传下载通道
- MapReduce（基于SQL引擎）
- Graph
- 机器学习算法

pandas

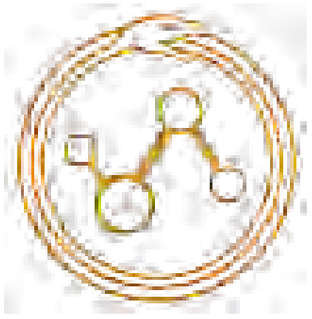
- 数据分析
- 基于numpy
- DataFrame（R data.frame）
- 绘图
- 轻松和Python众多三方库交互（matplotlib、jupyter）



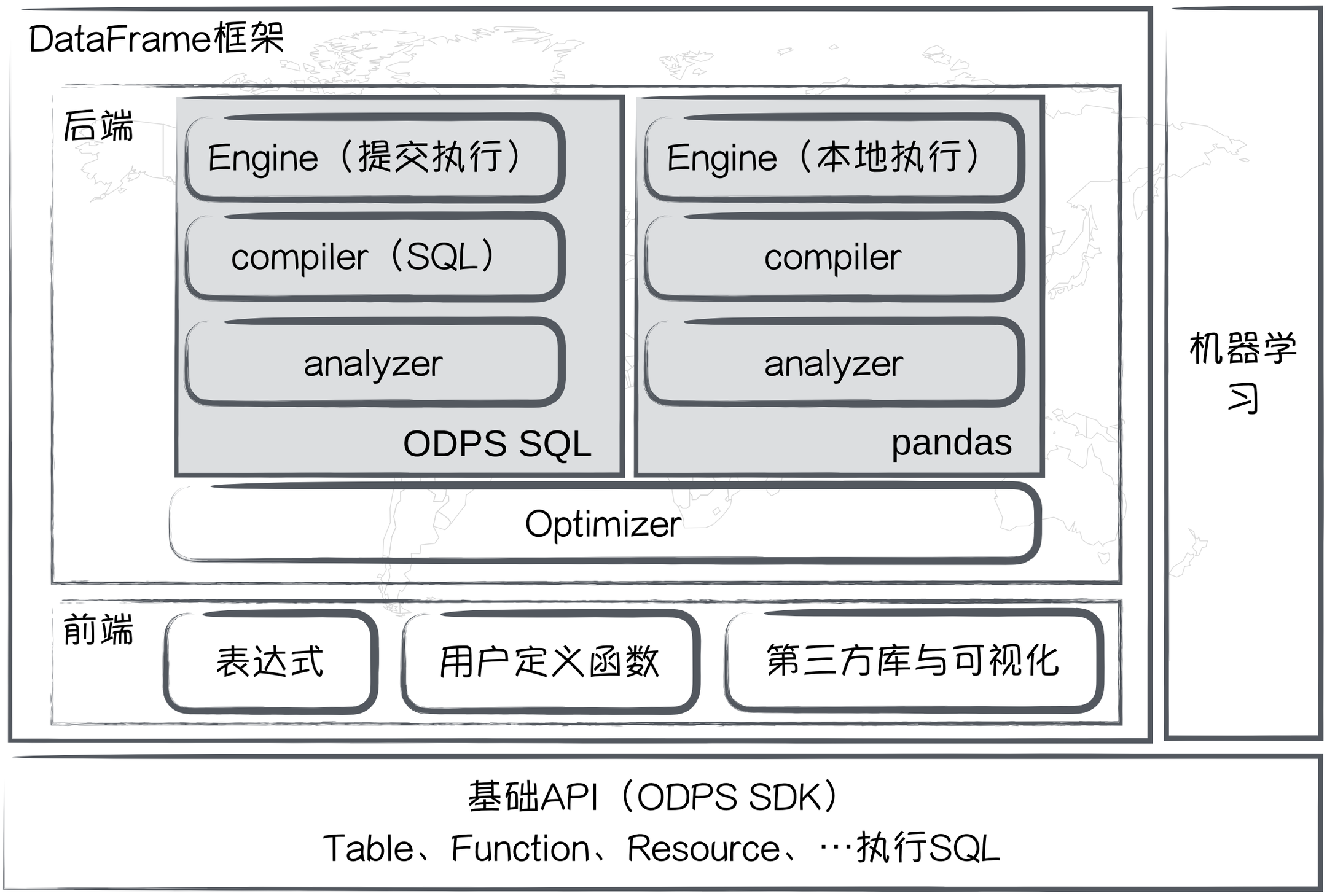
PyODPS

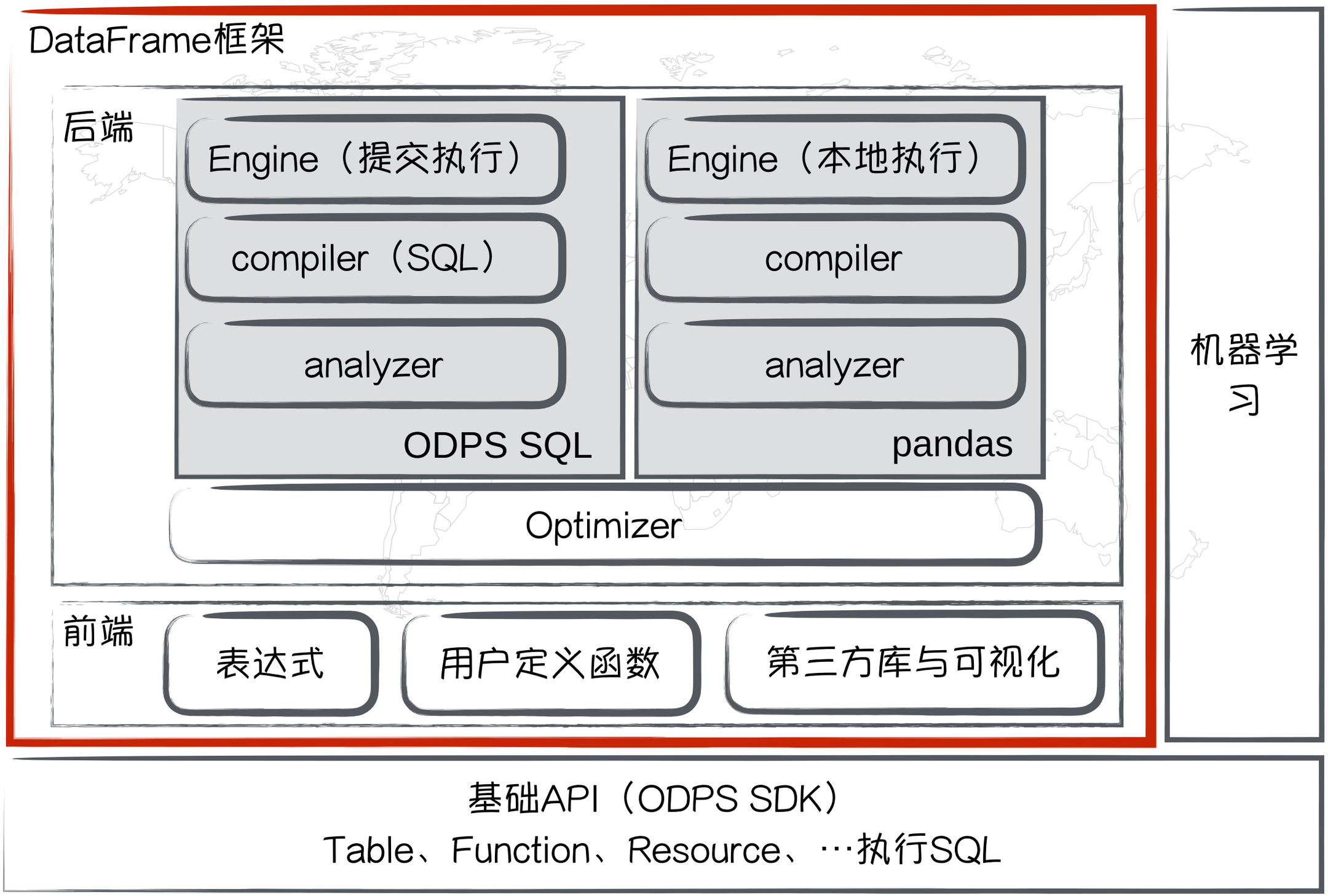
- https://github.com/aliyun/aliyun-odps-python-sdk
- docs：http://pyodps.readthedocs.org

- 不只是MaxCompute SDK
- **用Python优雅地处理大数据**
- DataFrame框架（pandas语法+多后端）
- ML机器学习（DataFrame输入输出）



PyODPS架构


DataFrame框架
后端
Engine（提交执行）
compiler（SQL）
analyzer
ODPS SQL
Engine（本地执行）
compiler
analyzer
pandas
Optimizer
前端
表达式
用户定义函数
第三方库与可视化
机器学习
基础API（ODPS SDK）
Table、Function、Resource、…执行SQL


DataFrame框架
后端
Engine（提交执行）
compiler（SQL）
analyzer
ODPS SQL
Engine（本地执行）
compiler
analyzer
pandas
Optimizer
前端
表达式
用户定义函数
第三方库与可视化
机器学习
基础API（ODPS SDK）
Table、Function、Resource、…执行SQL

背景

API、语法、编程语言

表达力

简单，符合直觉

存储、元数据

数据

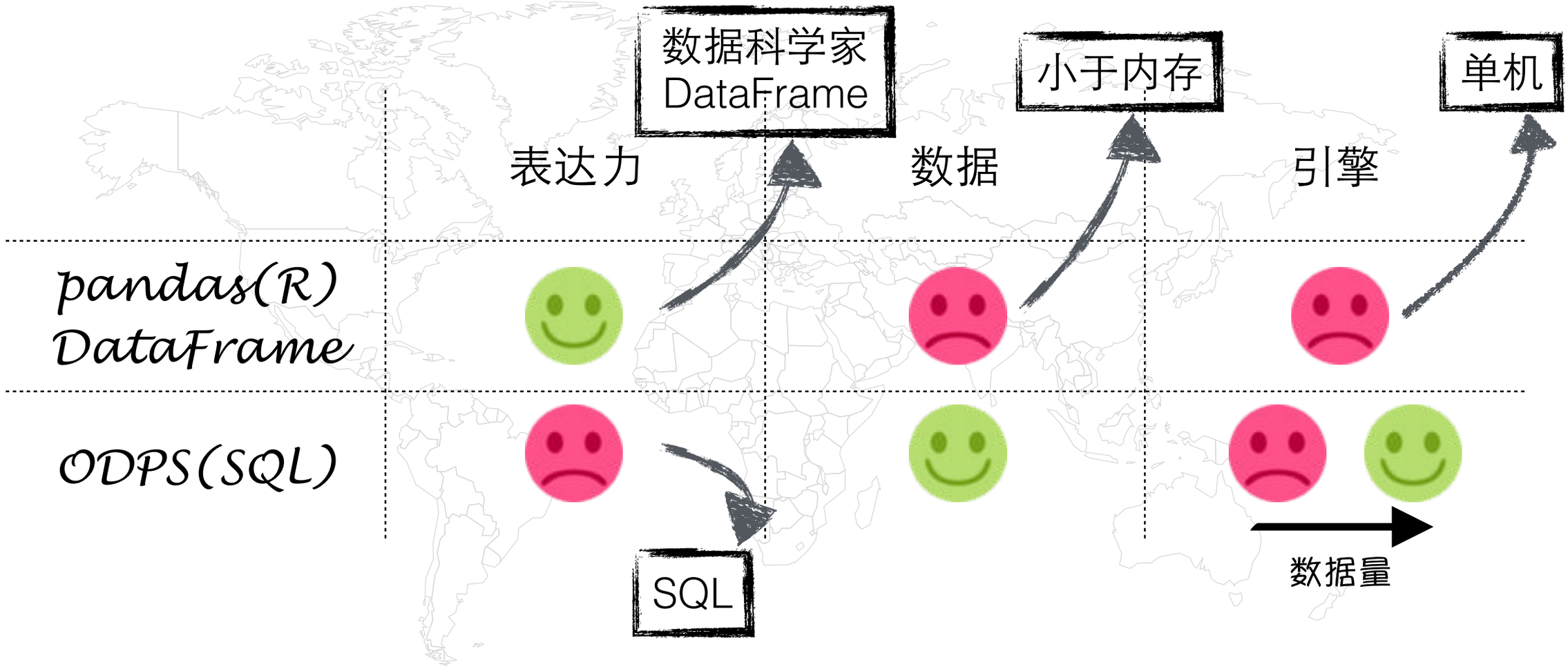
压缩、效率

计算

引擎

性能

背景



PyODPS DataFrame尝试综合两者的优点

PyODPS DataFrame

- Python语言，变量、条件判断、循环
- pandas-like API，自定义前端
- 后端：
  - visitor设计模式
  - ODPS SQL + pandas
  - 可扩展（spark）
- 延迟执行

API（pandas-like）

列选择
```
iris[‘sepal_width’, ‘sepal_length’]
```

过滤
```
iris[(iris.species == ‘Iris-setosa’) &
     (iris.sepal_length > 5.0)]
```

计算
```
(iris.sepal_length * 10).log()
```

自定义函数
```
iris.sepal_length.map(lambda x: x + 1)
```

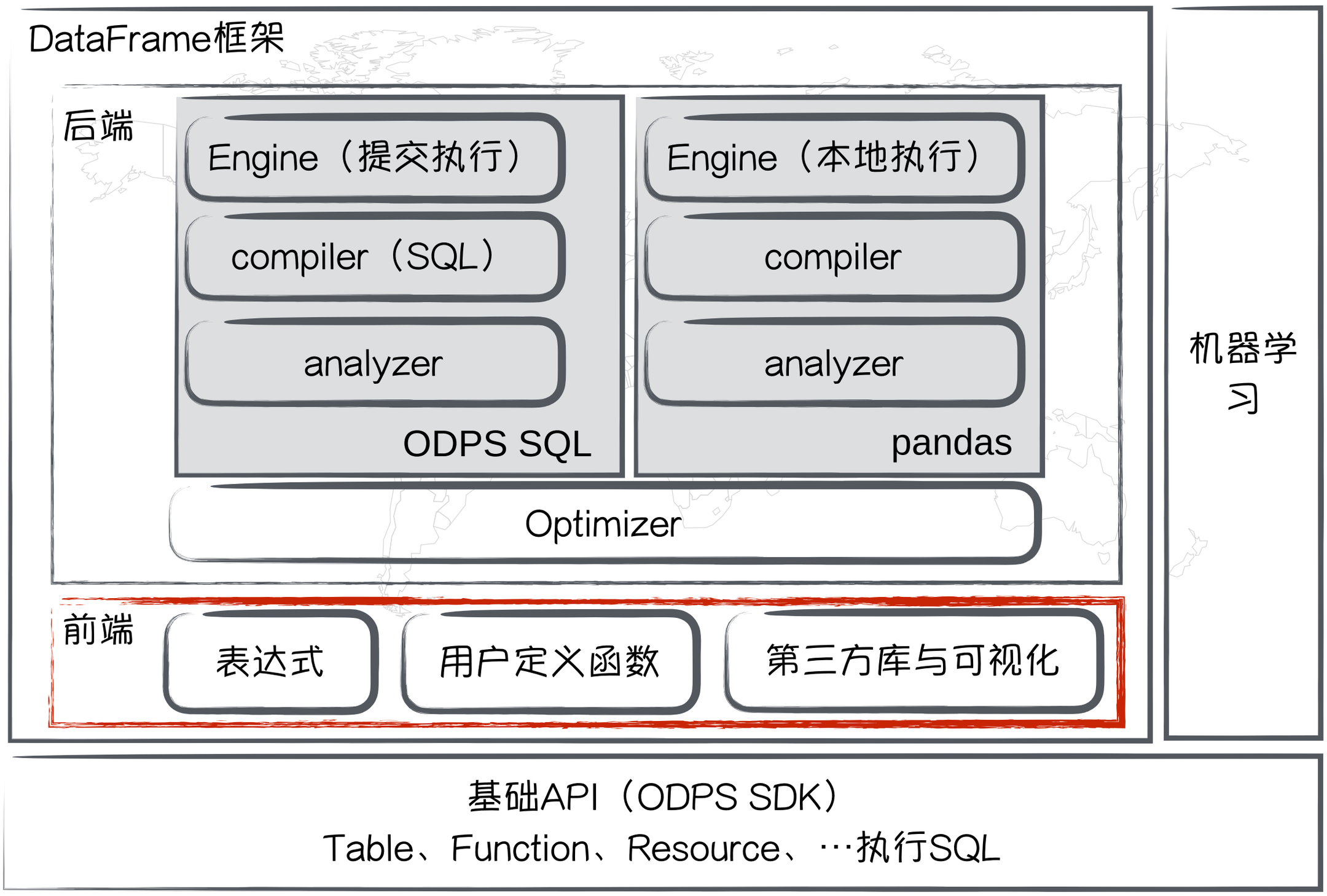
聚合
```
iris.sepal_length.mean()
```

分组聚合
```
iris.groupby(‘species’).agg(
    shortest=iris.petal_length.min(),
    longest=iris.petal_length.max(),
    average=iris.petal_length.mean())
```

…

Join, MapReduce…

PyODPS


DataFrame框架
后端
Engine（提交执行）
compiler（SQL）
analyzer
ODPS SQL
Engine（本地执行）
compiler
analyzer
pandas
Optimizer
前端
表达式
用户定义函数
第三方库与可视化
机器学习
基础API（ODPS SDK）
Table、Function、Resource、…执行SQL

表达式

```
iris.species.mean()
```

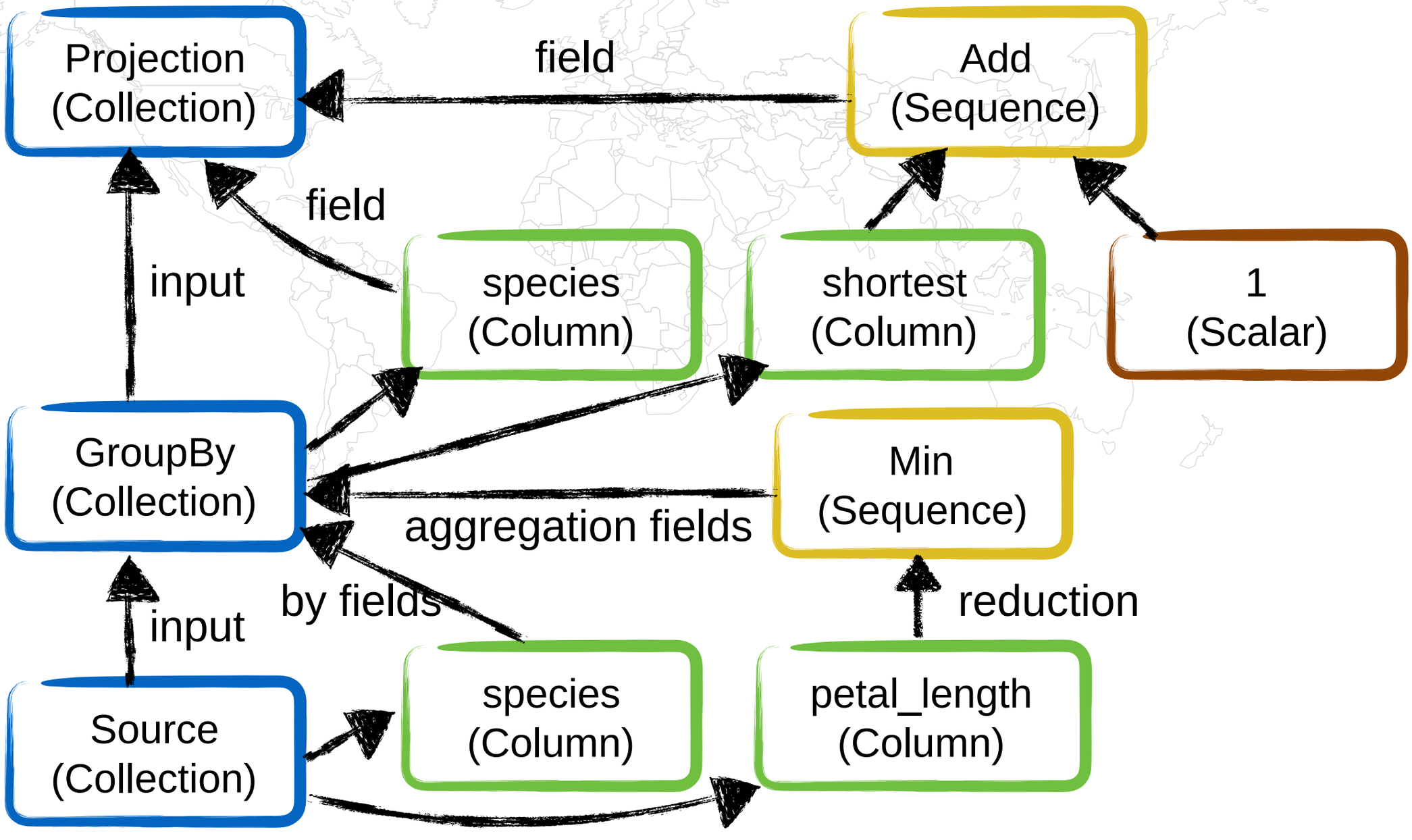
species是string类型，无法执行mean操作
表达式构建报错，无需等到运行时

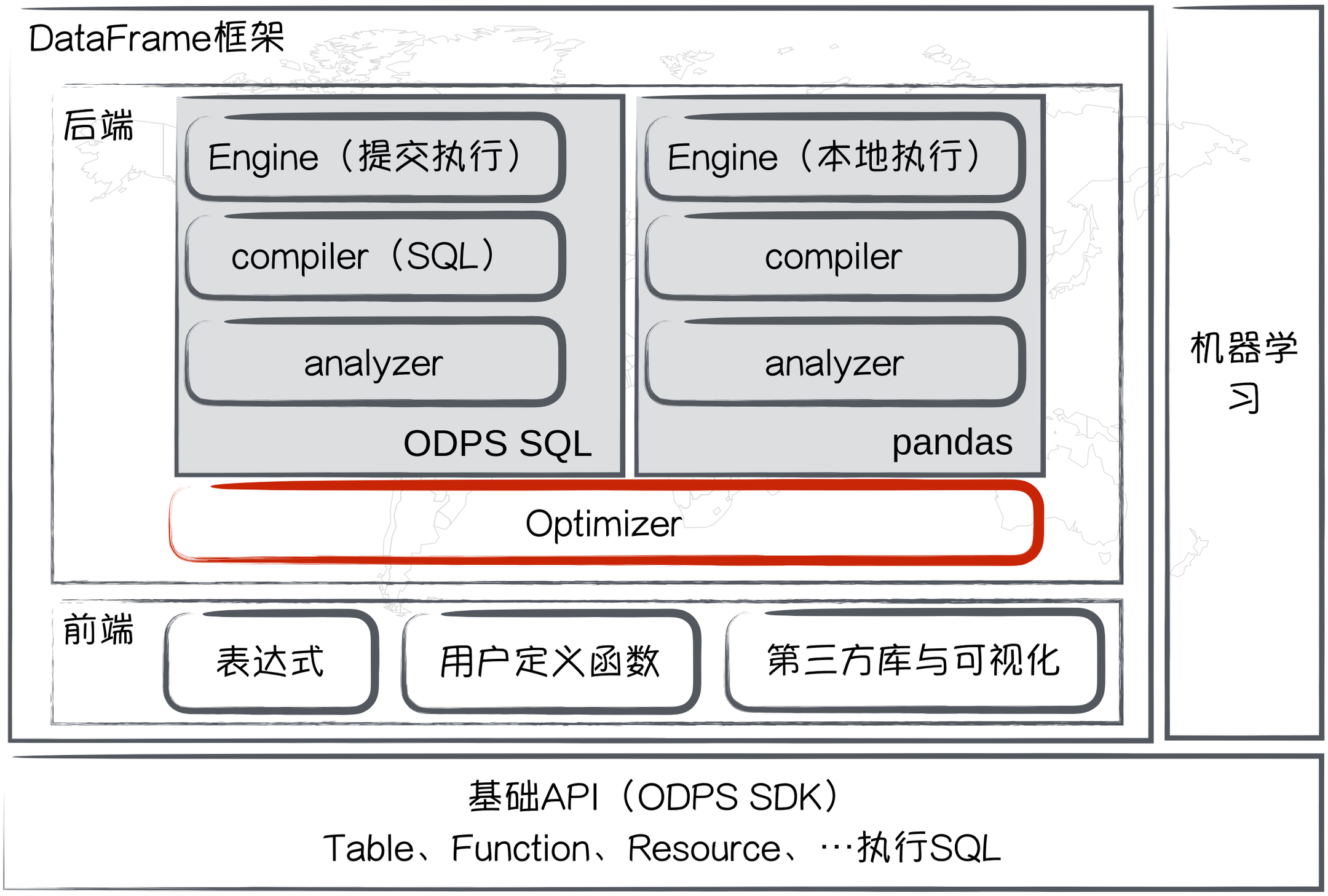
```
iris['not_exist_field']
```

不能存在的字段，同样表达式构建报错

表达式和抽象语法树

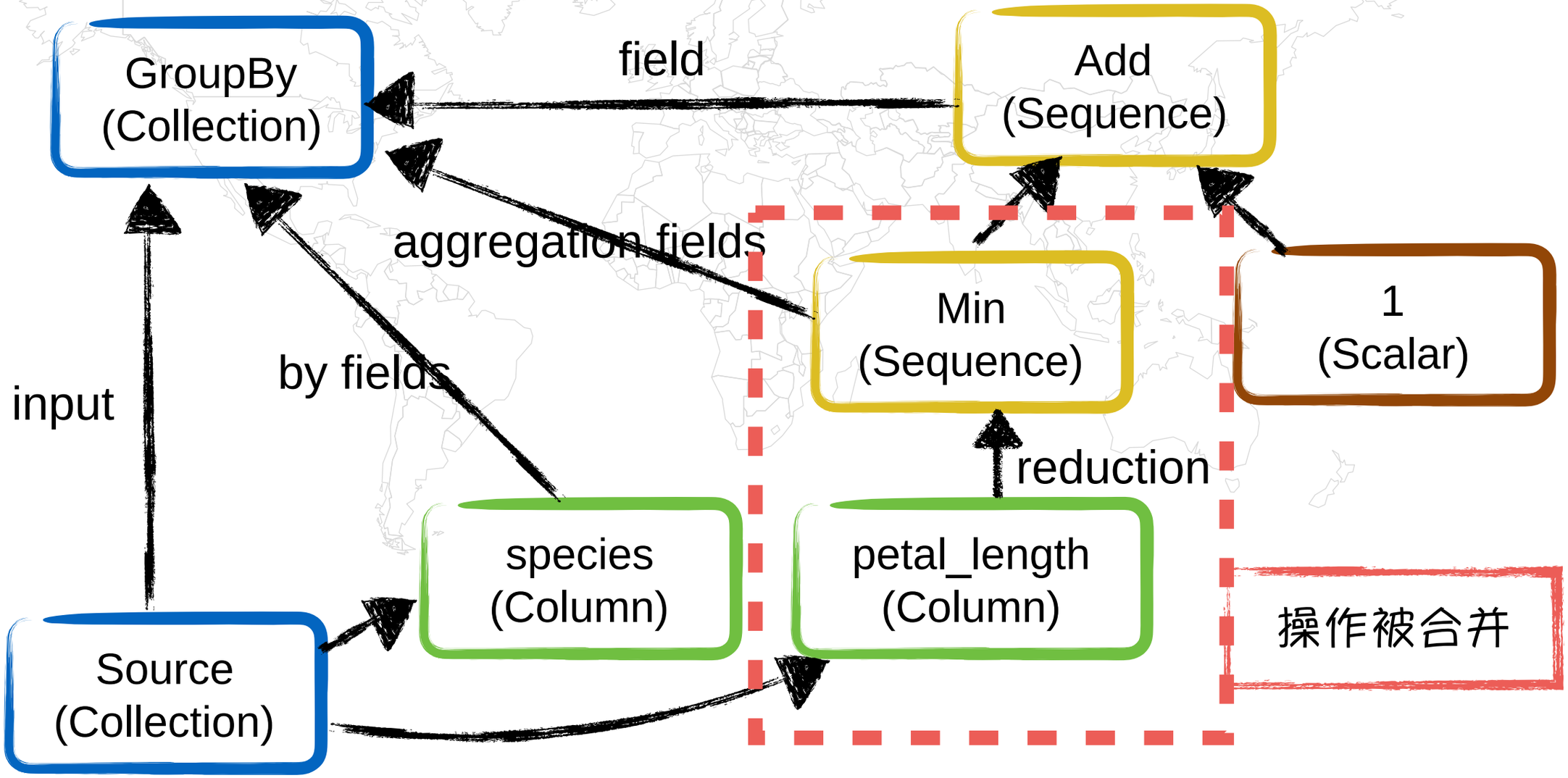
```
df = iris.groupby('species').agg(shortest=iris.petal_length.min())
df = df['species', df.shortest + 1]
```




DataFrame框架
后端
Engine（提交执行）
compiler（SQL）
analyzer
ODPS SQL
Engine（本地执行）
compiler
analyzer
pandas
Optimizer
前端
表达式
用户定义函数
第三方库与可视化
机器学习
基础API（ODPS SDK）
Table、Function、Resource、…执行SQL

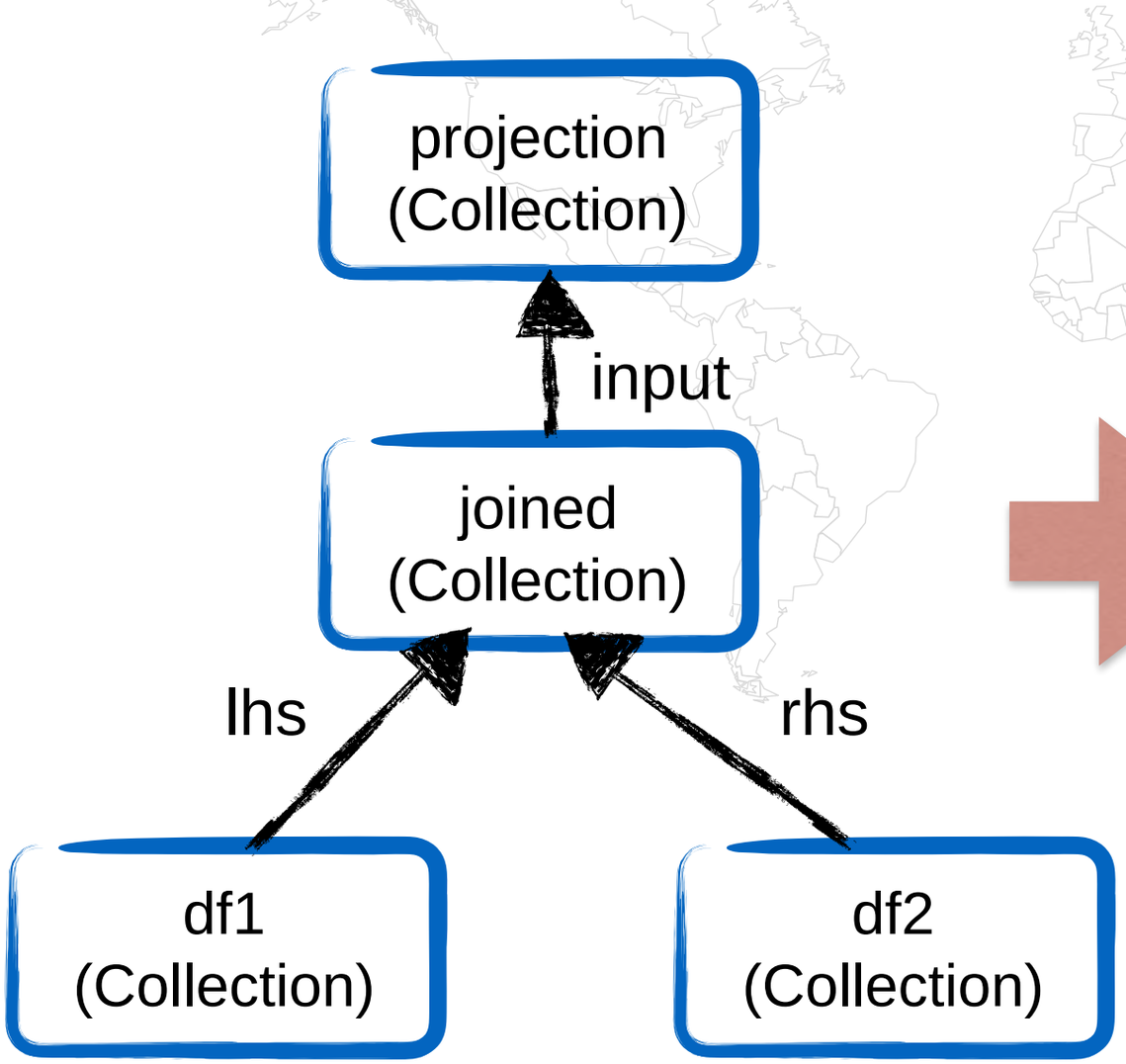
Optimizer（操作合并）

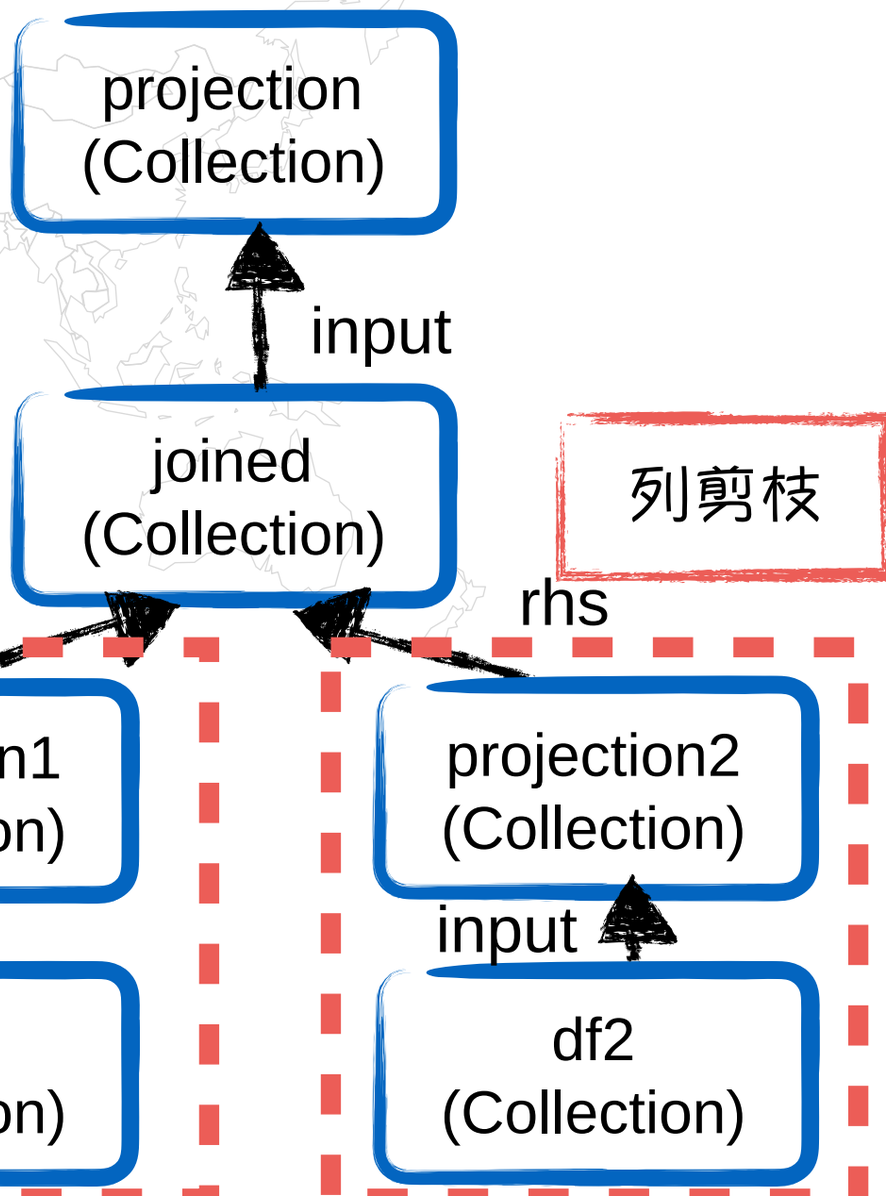
```
df = iris.groupby('species').agg(shortest=iris.petal_length.min())
df = df['species', df.shortest + 1]
```



Optimizer（列剪枝）

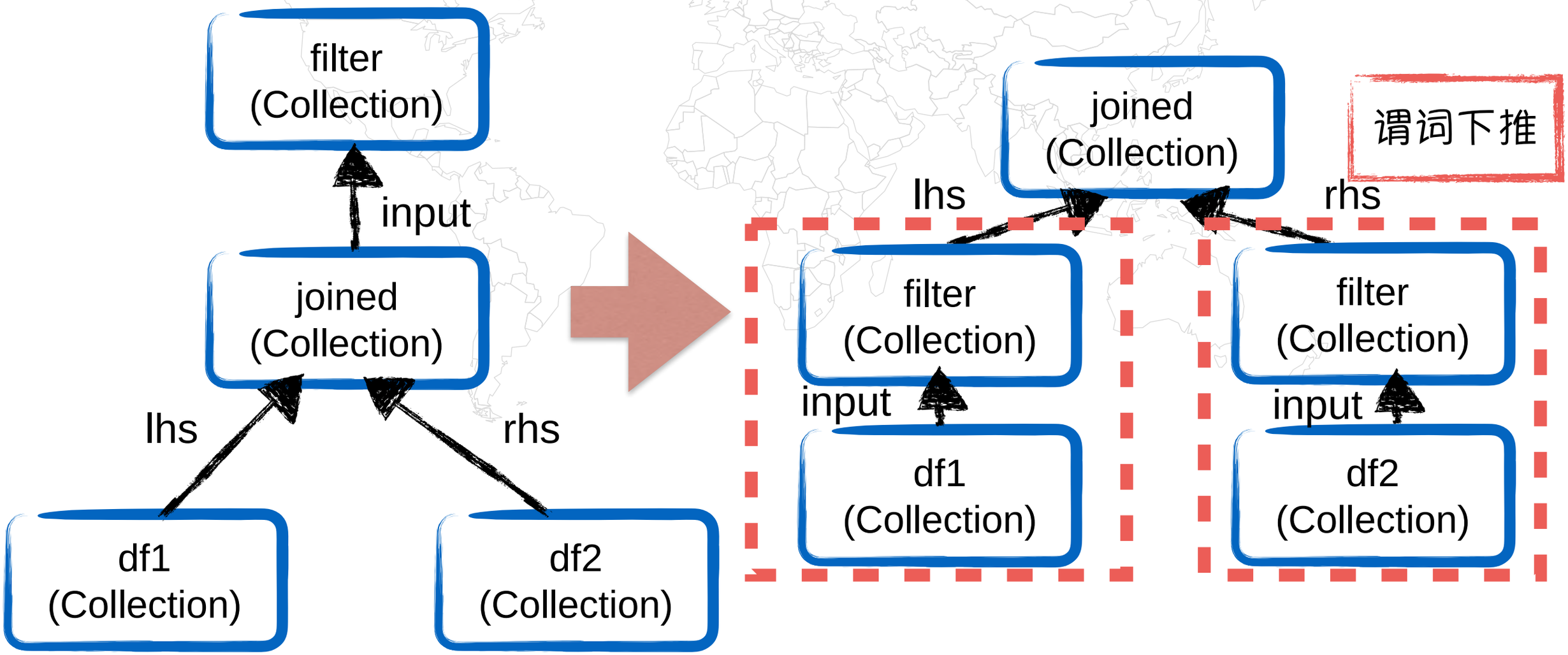
```
df = df1.join(df2)
df = df[‘field1_from_df1’, ‘field2_from_df2’]
```





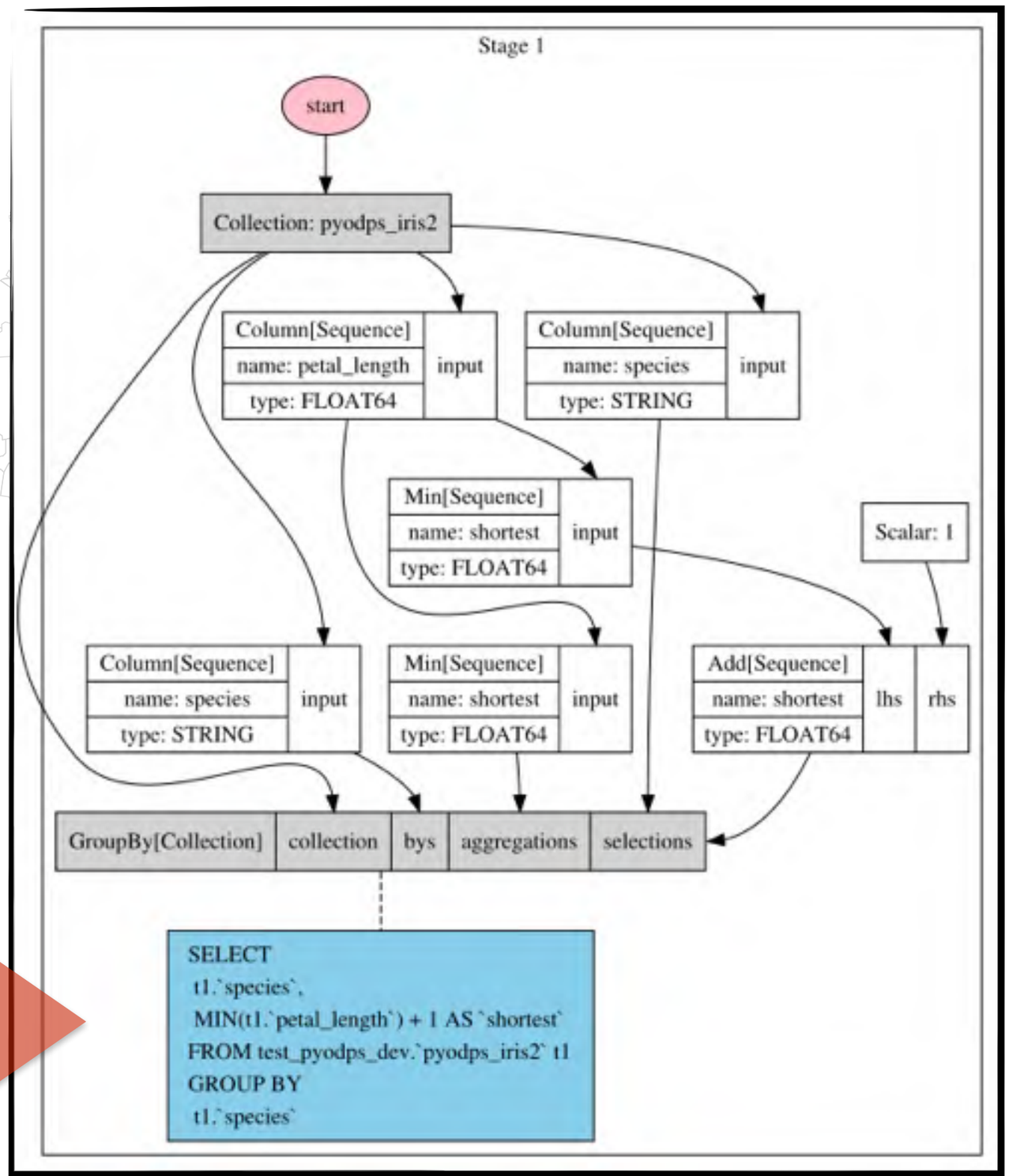
Optimizer（谓词下推）

```
df = df1.join(df2)
df = df.filter(df.field1_from_df1 < 10, df.field2_from_df2.len() > 5)
```

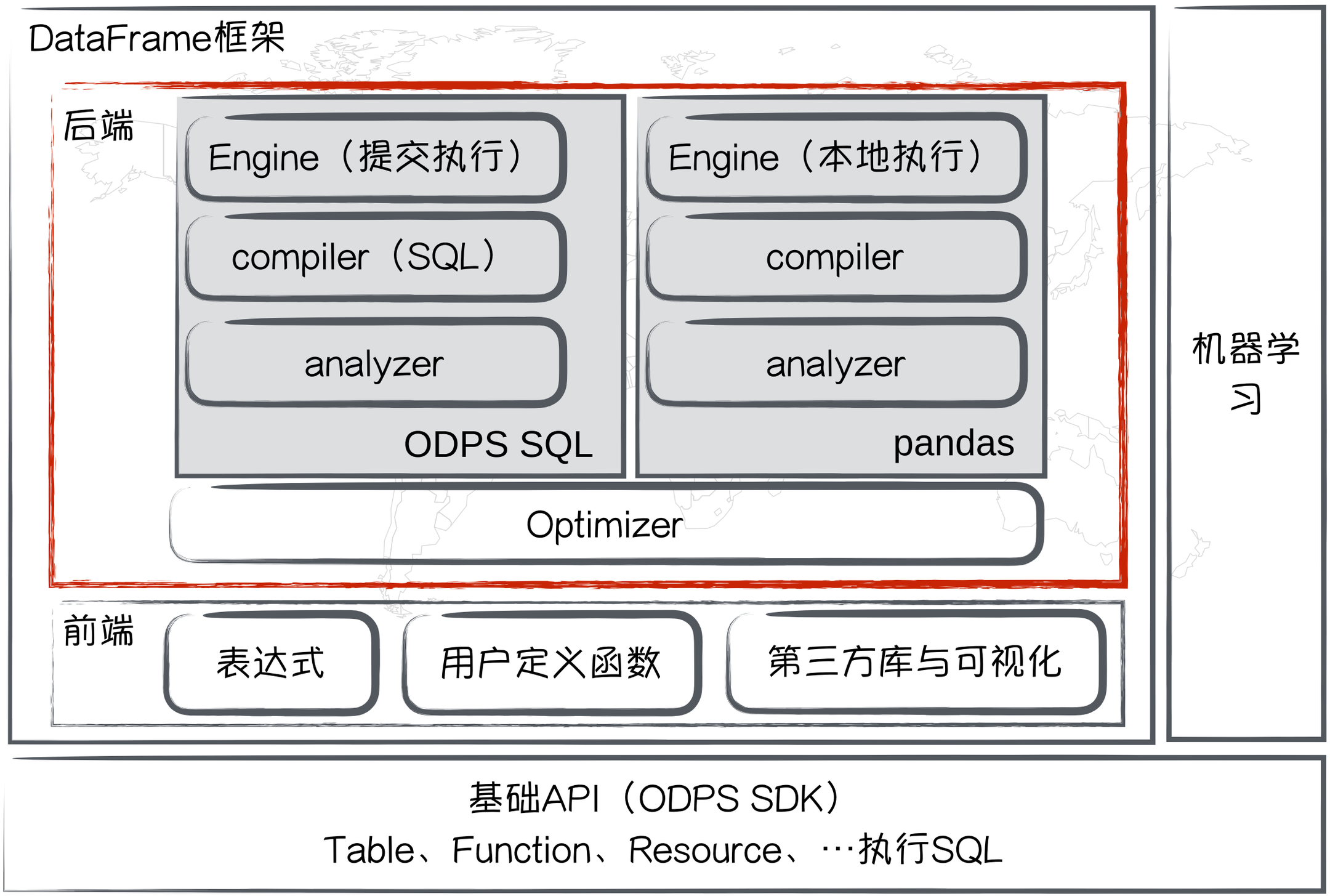


可视化

```
df = iris.groupby('species').agg(
    shortest=iris.petal_length.min())
df = df['species', df.shortest + 1]
df.visualize()
```

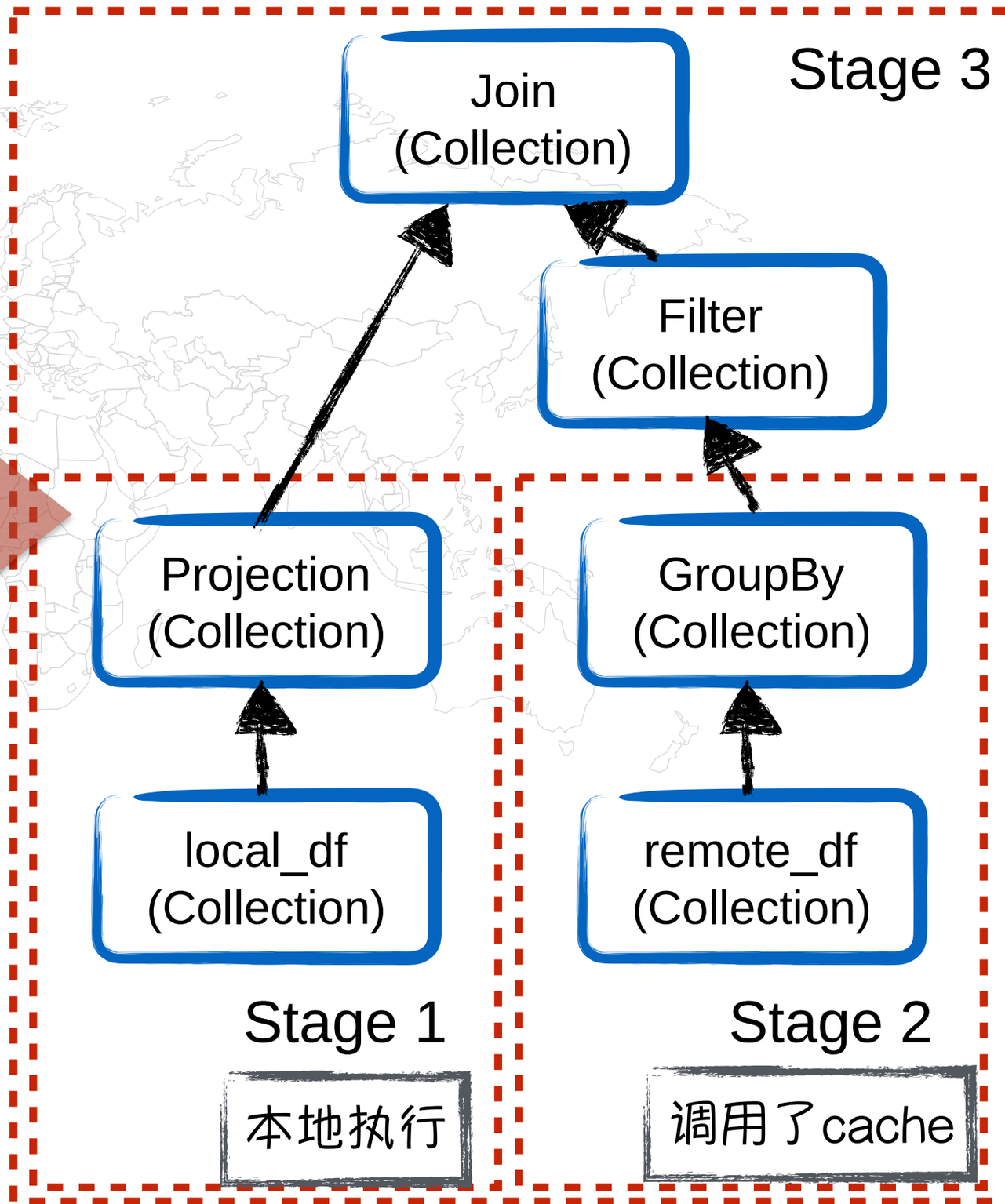


ODSP SQL后端会
compile成一条SQL执行

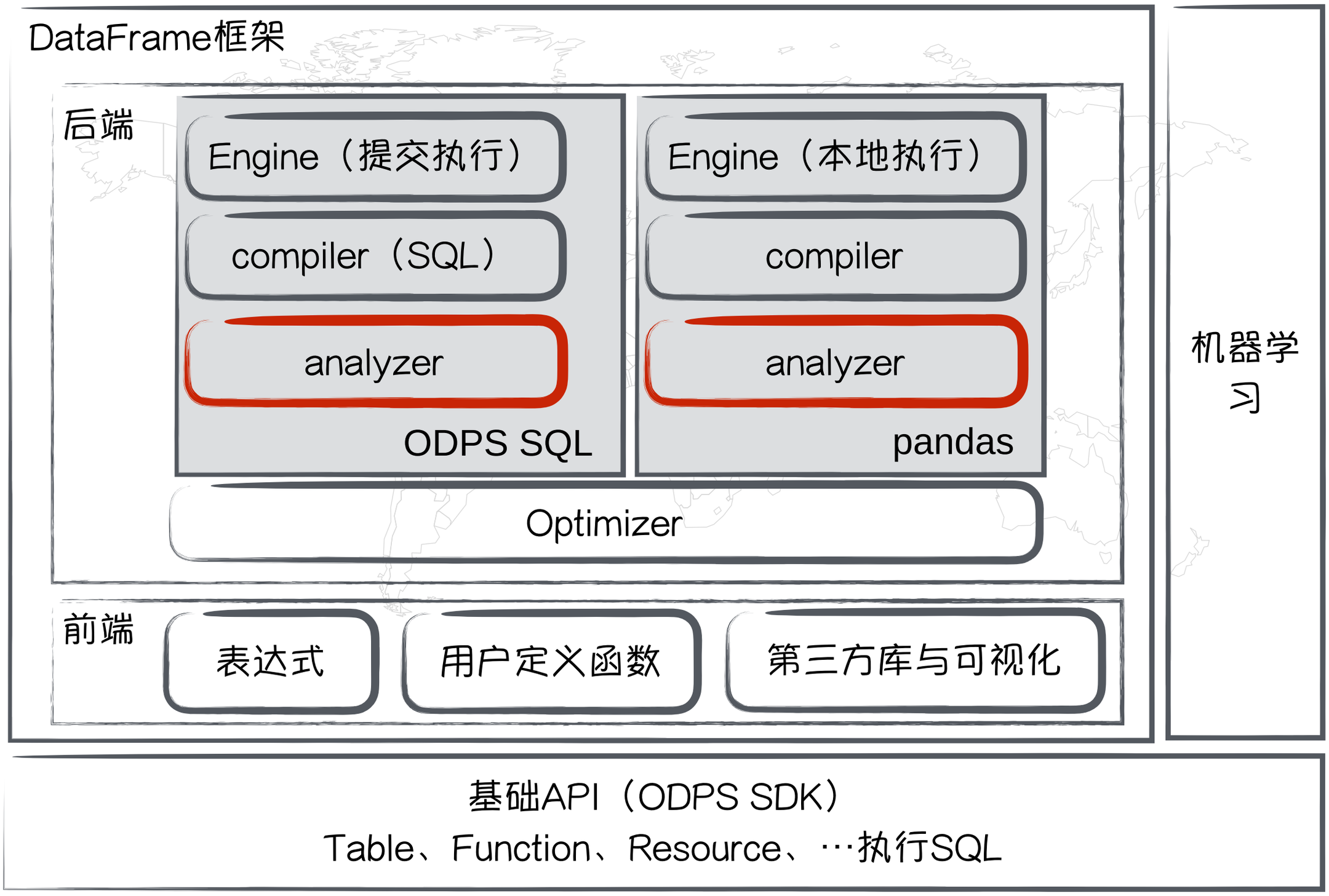

DataFrame框架
后端
Engine（提交执行）
compiler（SQL）
analyzer
ODPS SQL
Engine（本地执行）
compiler
analyzer
pandas
Optimizer
前端
表达式
用户定义函数
第三方库与可视化
机器学习
基础API（ODPS SDK）
Table、Function、Resource、…执行SQL

后端

```
local_df =
  read_csv('…').exclude(…)
remote_df =
  DataFrame(odps.get_table('my_table'))
remote_df2 =
  remote_df.groupby('field').agg(…).cache()
remote_df3 =
  remote_df2[remote_df2.field < 10]
df = local_df.join(remote_df3, on='id')
```



- Stage 1： pandas后端
- Stage 2, 3： ODPS SQL后端
- DAG按拓扑顺序执行

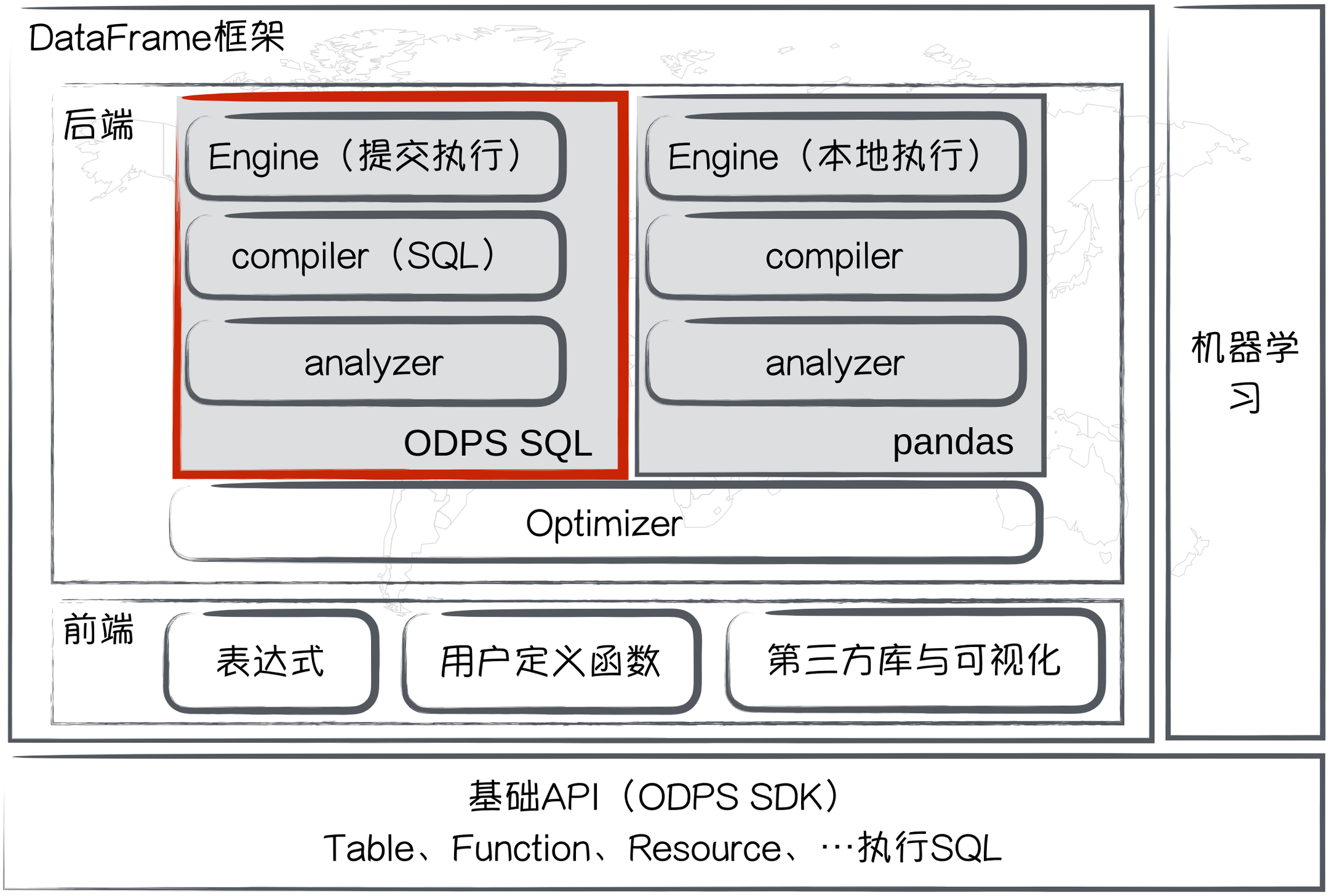

DataFrame框架
后端
Engine（提交执行）
compiler（SQL）
analyzer
ODPS SQL
Engine（本地执行）
compiler
analyzer
pandas
Optimizer
前端
表达式
用户定义函数
第三方库与可视化
机器学习
基础API（ODPS SDK）
Table、Function、Resource、…执行SQL

Analyzer

- Analyzer的作用是针对具体的后端，将一些操作进行转化

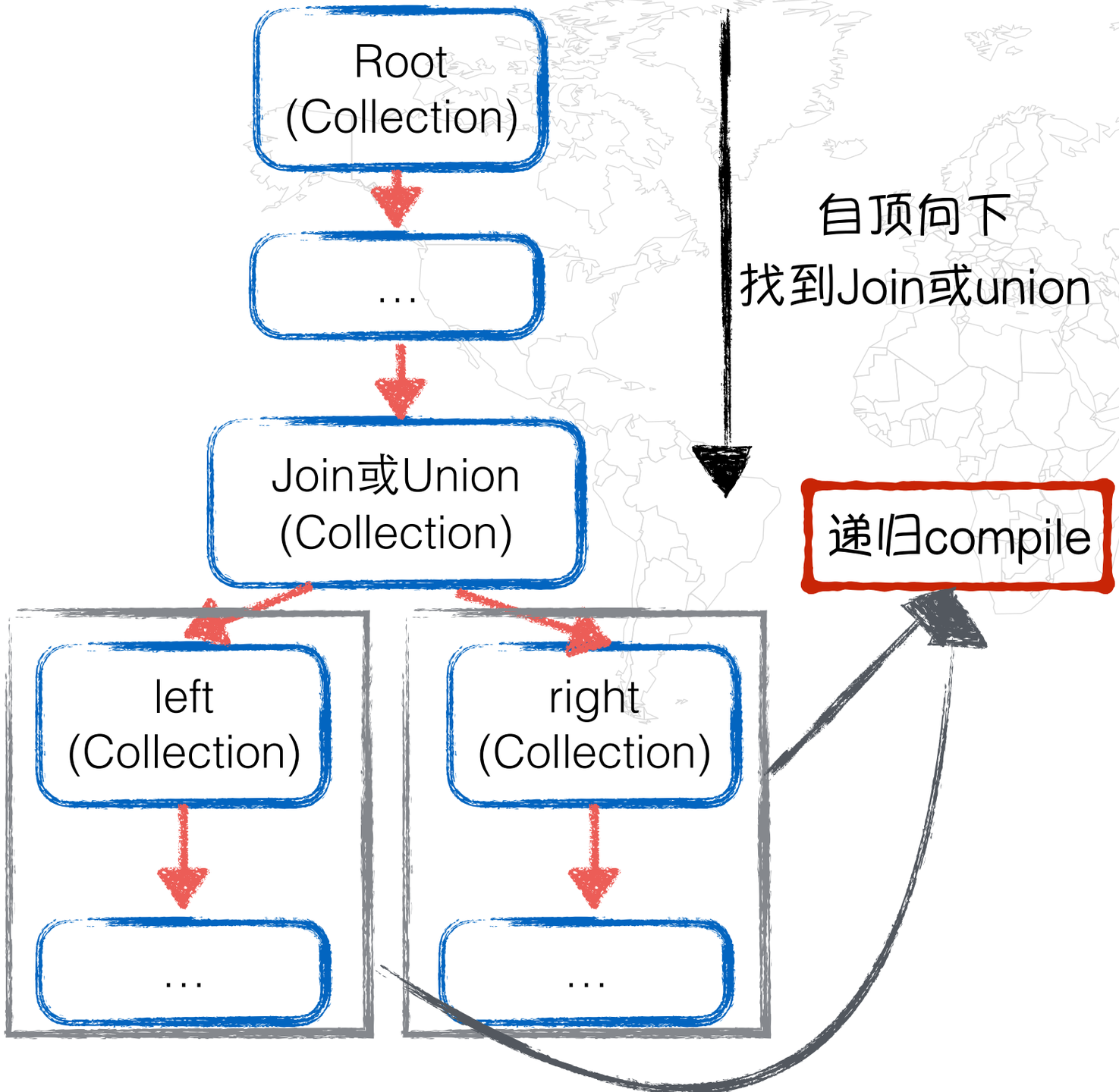
- 有些操作比如value_counts，pandas本身支持，因此对于pandas后端，无需处理；对于ODPS SQL后端，在analyzer执行的时候，会被改写成groupby+sort的操作

- 还有一些算子，在compile到ODPS SQL时，没有内建函数能完成，会被改写成自定义函数


DataFrame框架
后端
Engine（提交执行）
compiler（SQL）
analyzer
ODPS SQL
Engine（本地执行）
compiler
analyzer
pandas
Optimizer
前端
表达式
用户定义函数
第三方库与可视化
机器学习
基础API（ODPS SDK）
Table、Function、Resource、…执行SQL

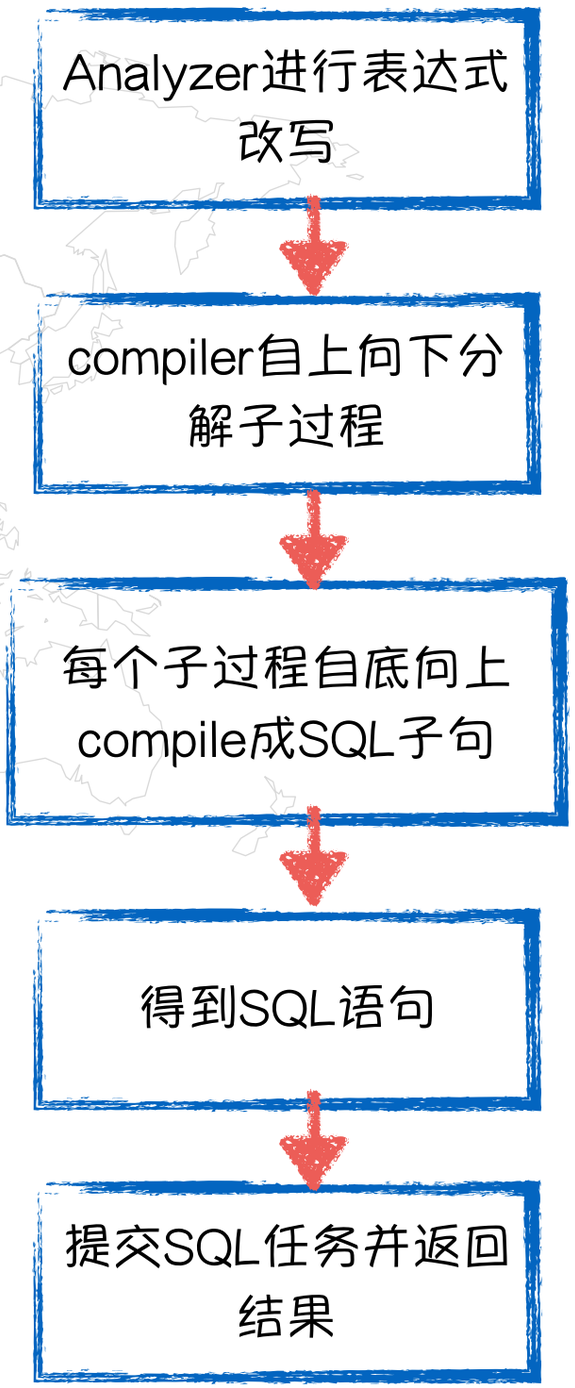
ODPS SQL后端

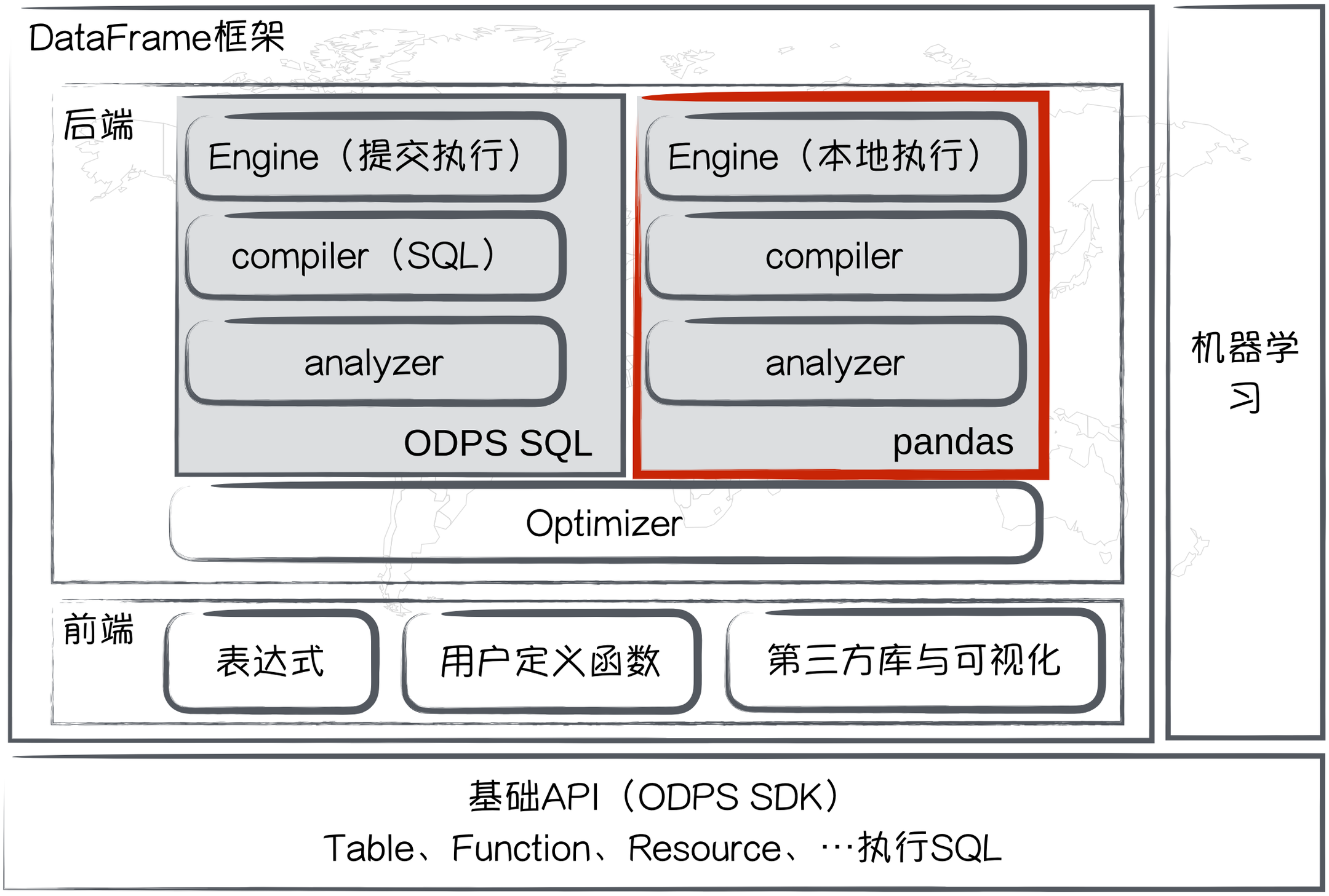
Compiler



自顶向下
找到Join或union

Engine




DataFrame框架
后端
Engine（提交执行）
compiler（SQL）
analyzer
ODPS SQL
Engine（本地执行）
compiler
analyzer
pandas
Optimizer
前端
表达式
用户定义函数
第三方库与可视化
机器学习
基础API（ODPS SDK）
Table、Function、Resource、…执行SQL

pandas后端

- compiler：将每个表达式操作转换成pandas操作，最后形成DAG，
- engine：执行按DAG拓扑顺序执行
- 作用：
  - 本地DEBUG
  - 数据量较小时可以使用pandas计算

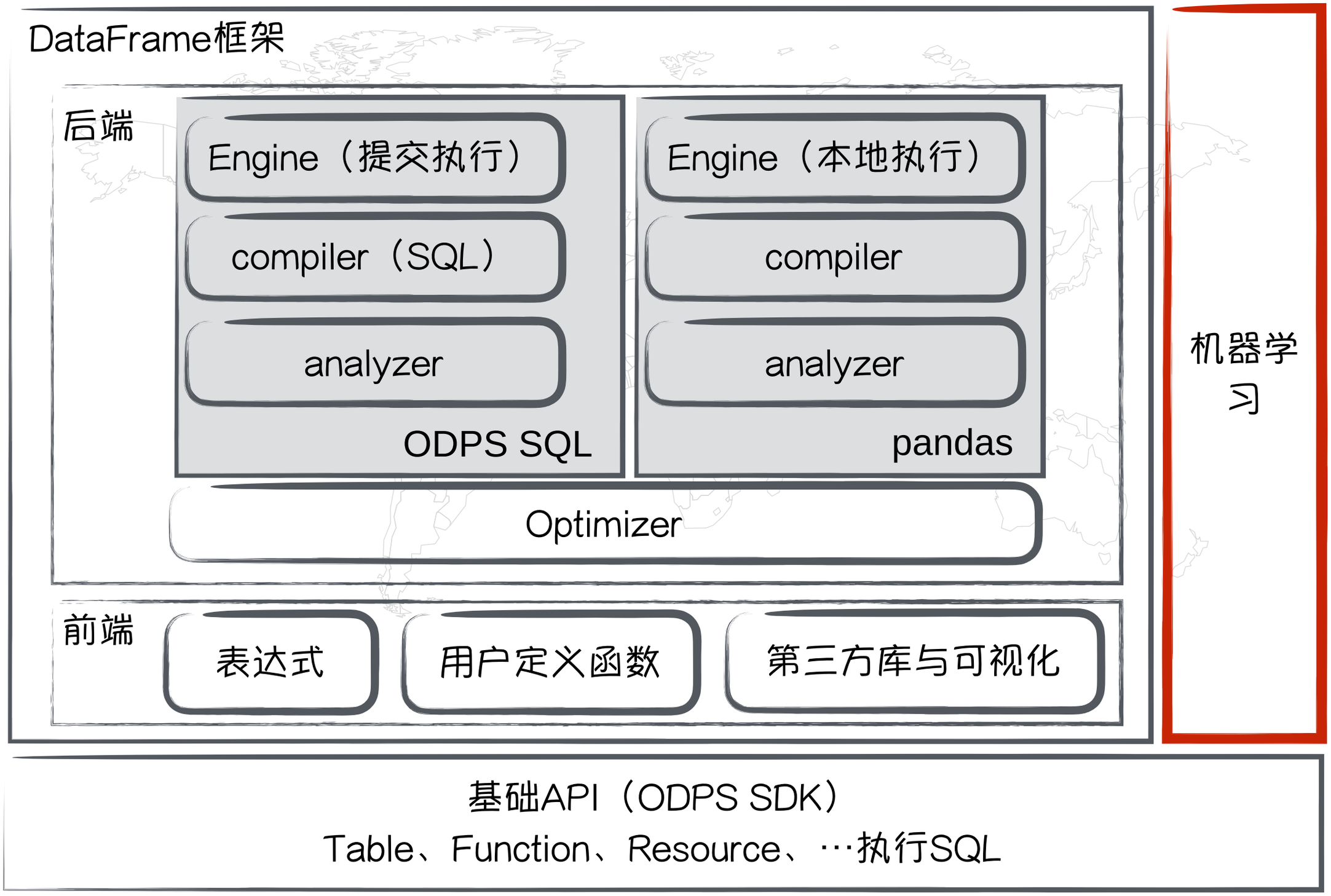
难点+坑

- 后端编译出错容易丢失上下文
  - 多次optimize和analyze，导致难以查出是之前哪处visit node导致
    - 解决：保证每个模块独立性、测试完备
- bytecode兼容问题
- SQL的执行顺序

```
df1 = df.select(df, a2=df.a+1)
df2 = df1.filter(df1.a2 < 10)
```

SQL compiled

```
SELECT *
FROM (
  SELECT
    t1.`a`,
    t1.`b`,
    t1.`c`,
    t1.`d`,
    t1.`e`,
    t1.`a` + 1 AS `a2`
  FROM test_pyodps_dev.`pyodps_test_seq` t1
) t2
WHERE t2.`a2` < 10
```


DataFrame框架
后端
Engine（提交执行）
compiler（SQL）
analyzer
ODPS SQL
Engine（本地执行）
compiler
analyzer
pandas
Optimizer
前端
表达式
用户定义函数
第三方库与可视化
机器学习
基础API（ODPS SDK）
Table、Function、Resource、…执行SQL

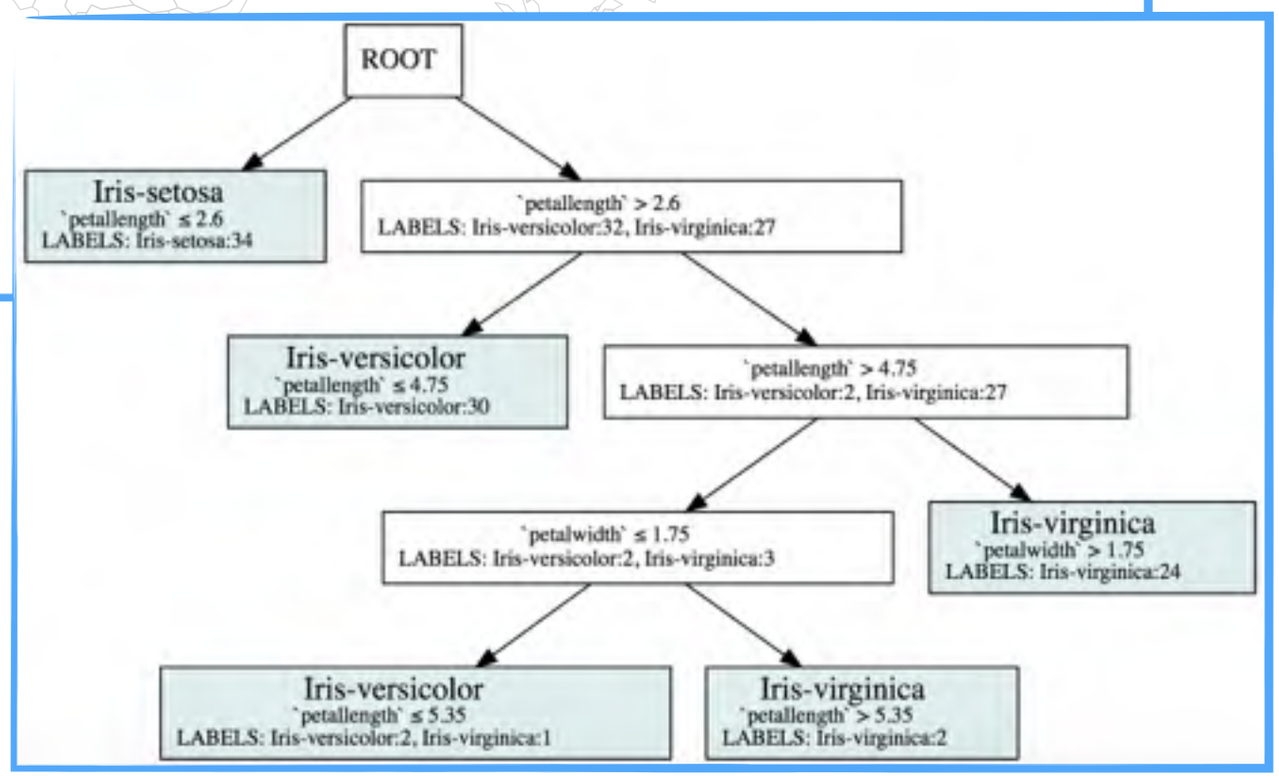
ML机器学习

```
from odps.ml.classifiers import RandomForests

df = iris.roles(label='name')
train, test = df.split(0.6)

model = RandomForests(tree_num=1)
trained = model.train(train)
predicted = trained.predict(test)

trained.segments[0]
```



与jupyter notebook集成

```
In [1]: %load_ext odps
In [2]: %enter
```

```
In [*]: iris[iris.sepal_width < 3.5].head(10)
```

(0.00%)

Start to execute sc

jupyter promotion Last Checkpoint: a day ago (unsaved changes)

File Edit View Insert

Python 2

In [13]: pd_df = pd.D

In [15]: %persist pd_

Progress of "DataFrame Operation[Filter] execution details"

Generate time: 2016-05-22 23:15:14

Instance 0: Running

AnonymousSQLTask: SUCCESS

M1_Stg1_job0

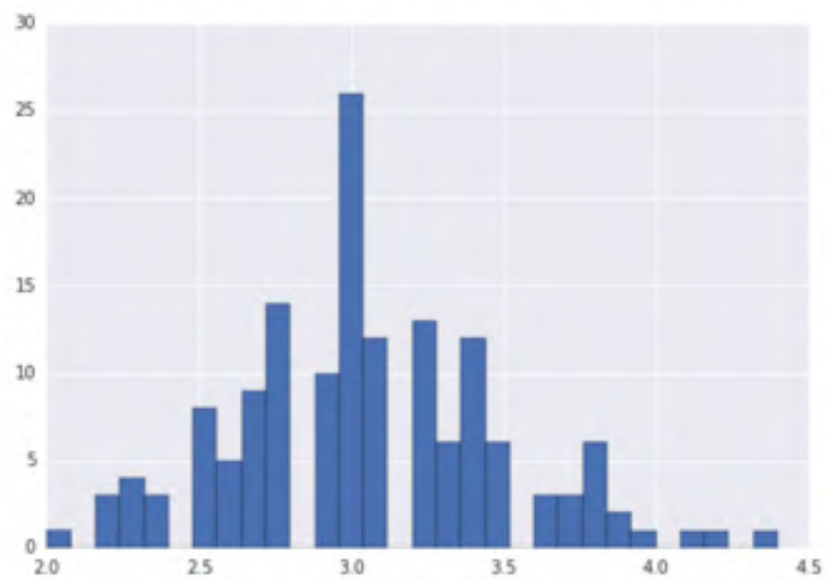
Logview

Close

width < 3.5].head(10)

DataFrame Operation[Filter] execution details

```
In [6]: iris.sepal_width.hist(bins=30)
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x113269b90>
```



PyODPS

DataFrame execution succeeded

localhost:8888

与jupyter notebook集成

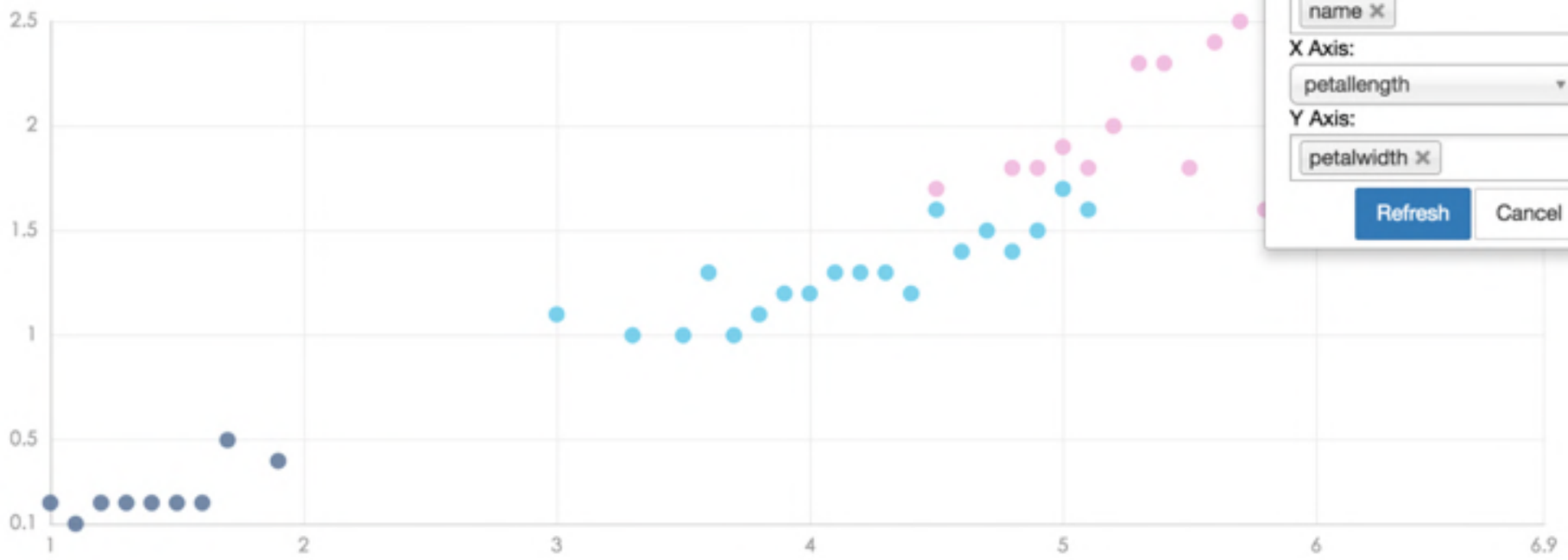
```
In [6]: iris = DataFrame(o.get_table('pyodps_iris'))

In [7]: iris
```

sepallength	sepalwidth	petallength	petalwidth	name
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6				
5				
5.4				
4.6				
5				
4.4				
4.9				

« 1 2 3 4

```
In [4]: iris
```



未来计划

- 分布式numpy，DataFrame基于分布式numpy的后端
- 内存计算，提升交互式体验
- tensorflow

Demo

Q&A

chinekingseu

秦续业

18616109836