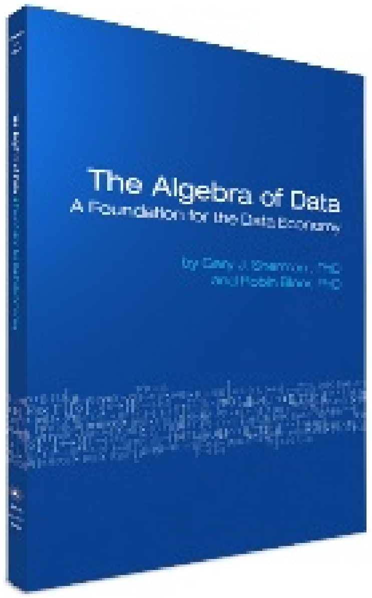
SPARK SUMMIT 2017

JUST-IN-TIME ANALYTICS AND THE NEED FOR AUTONOMOUS DATABASE ADMINISTRATION

Wes Holler, *Chief Architect*

Algebraix Data

Data Algebra



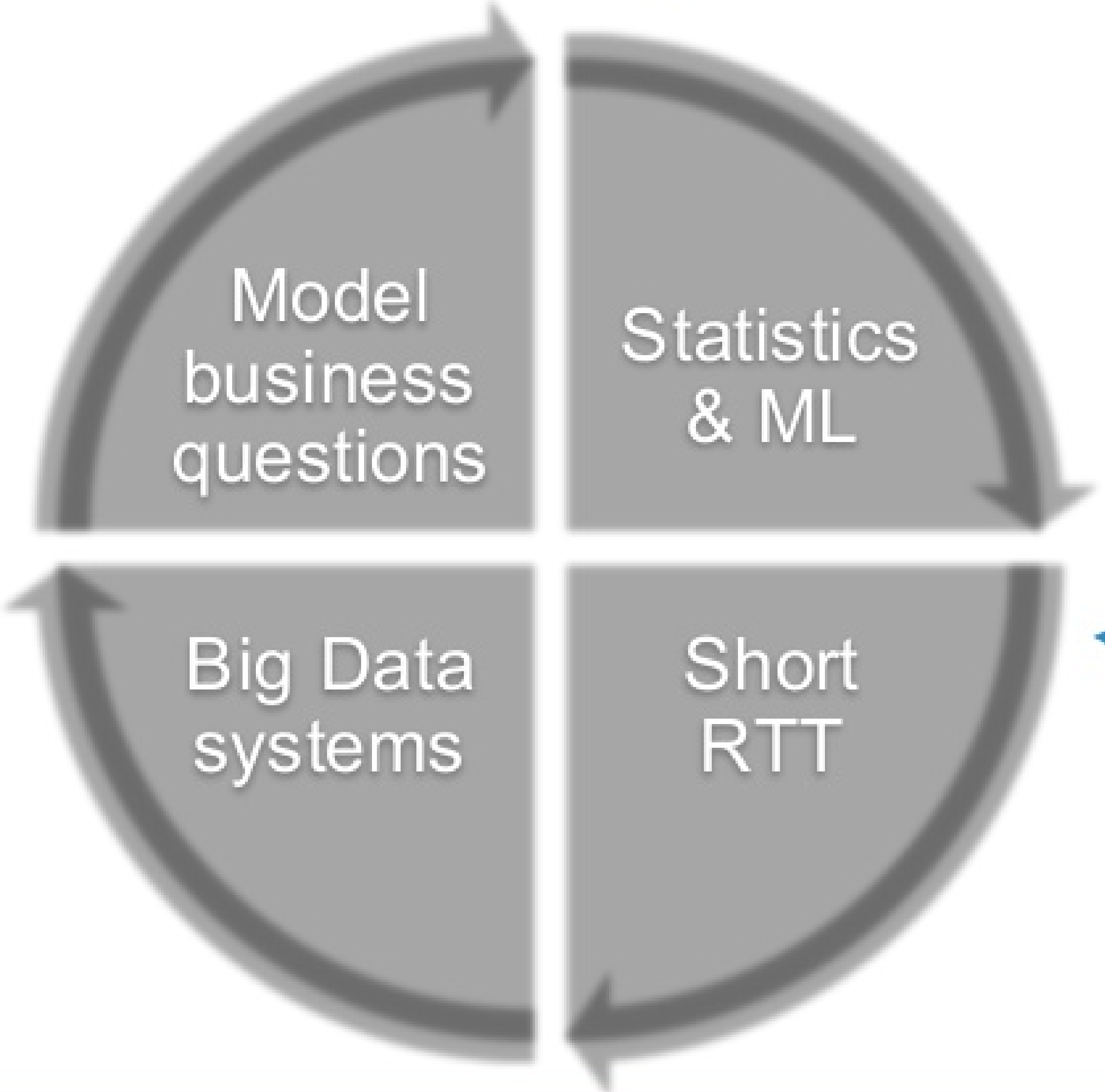
- A **novel application of Set Theory** to Data Processing
- **Applicable to many data models** including SQL

Just-in-Time Analytics

needs…

Autonomous Data Management

JIT Analytics and the Life of a Modern Analyst



This shouldn't require intimate knowledge of how underlying systems work.

Spark for JIT Analytics: *The Good*

- Unified API
- Schema-on-read and Heterogeneous Data Sources
- Declarative Languages/APIs and Catalyst
- Elastic Compute

Spark for JIT Analytics: *The Bad*

- Challenges for interactivity, efficiency, and scalability
- Cost of creating and maintaining "glue code"
- ***Data scientists and engineers are doing DBA work***

Database Management Responsibilities

Capacity planning

Configuration

We will focus on the performance and tuning aspects

Performance tuning

A billion other things

Improving and Maintaining Performance

- Indexes
- Materialize views
- Pre-aggregate data
- *Lots* of configuration

Performance Tuning Strategies in Spark

- Segment, cache, and checkpoint
- Configure cluster parameters
- `spark.sql.shuffle.partitions`

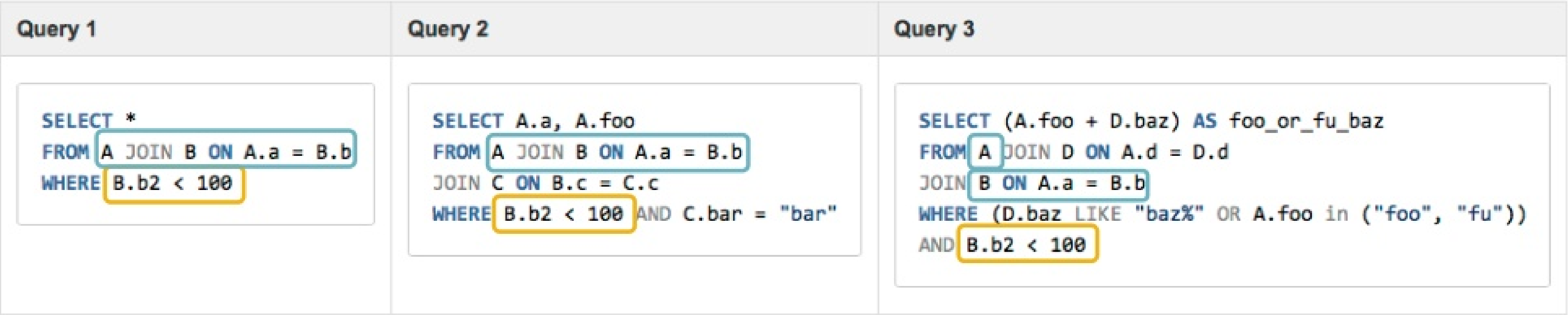
What is the Problem with Manual Tuning?

- Varies with the data (skew and scale), queries, and hardware
- Often done through trial and error
- Problems are exacerbated with JIT analytics case
- Shared resources

What is the Problem with Manual Tuning?

It is hard and time-consuming.

A Motivating Example for
Autonomous Data Management



Query 1

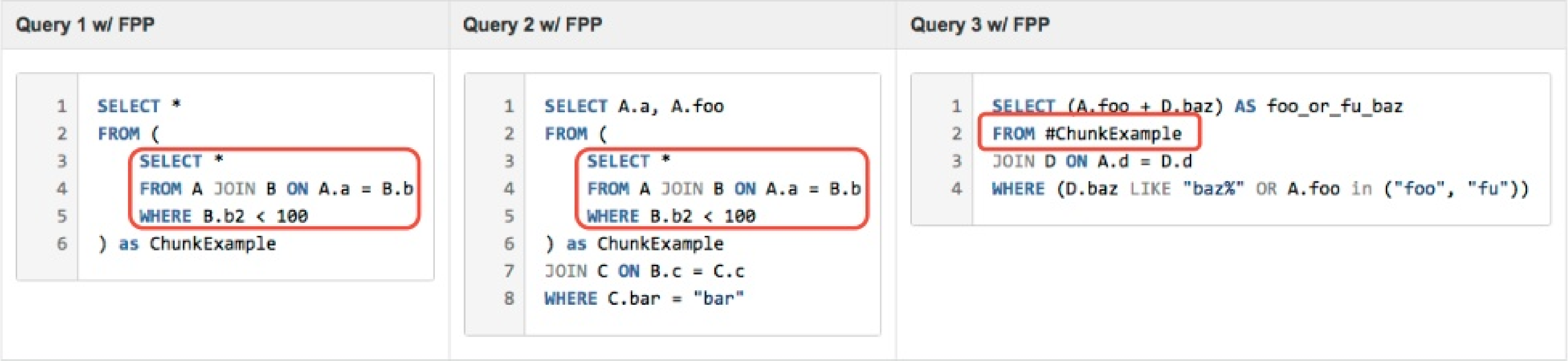
```
SELECT *
FROM A JOIN B ON A.a = B.b
WHERE B.b2 < 100
```

Query 2

```
SELECT A.a, A.foo
FROM A JOIN B ON A.a = B.b
JOIN C ON B.c = C.c
WHERE B.b2 < 100 AND C.bar = "bar"
```

Query 3

```
SELECT (A.foo + D.baz) AS foo_or_fu_baz
FROM A JOIN D ON A.d = D.d
JOIN B ON A.a = B.b
WHERE (D.baz LIKE "baz%" OR A.foo in ("foo", "fu"))
AND B.b2 < 100
```



Query 1 w/ FPP

```
SELECT *
FROM (
    SELECT *
    FROM A JOIN B ON A.a = B.b
    WHERE B.b2 < 100
) as ChunkExample
```

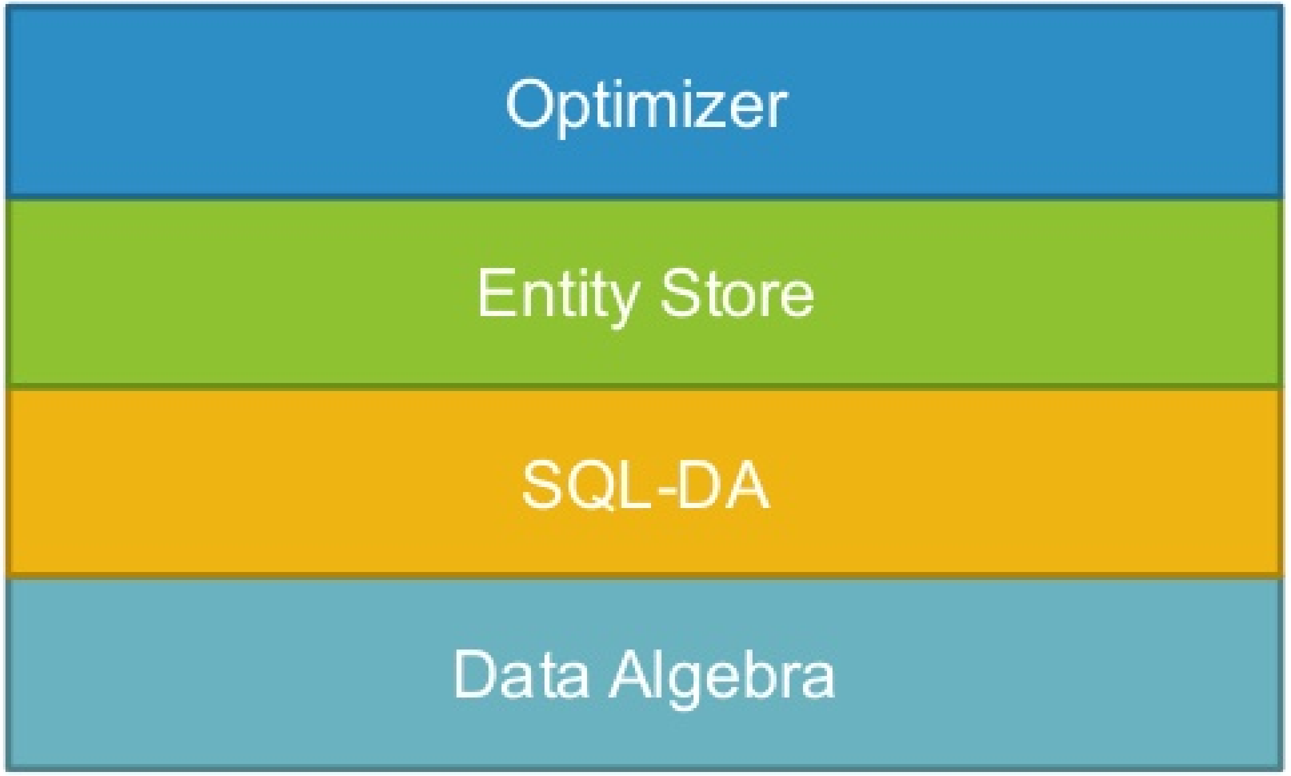
Query 2 w/ FPP

```
SELECT A.a, A.foo
FROM (
    SELECT *
    FROM A JOIN B ON A.a = B.b
    WHERE B.b2 < 100
) as ChunkExample
JOIN C ON B.c = C.c
WHERE C.bar = "bar"
```

Query 3 w/ FPP

```
SELECT (A.foo + D.baz) AS foo_or_fu_baz
FROM #ChunkExample
JOIN D ON A.d = D.d
WHERE (D.baz LIKE "baz%" OR A.foo in ("foo", "fu"))
```

A Motivating Example for Autonomous Data Management

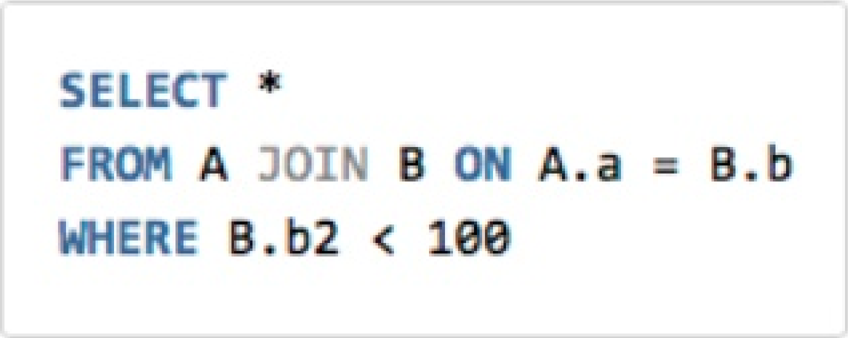


Optimizer

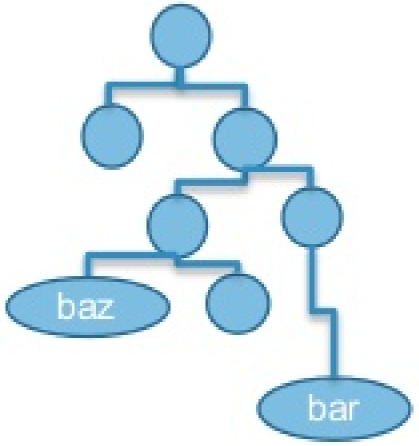
Entity Store

SQL-DA

Data Algebra

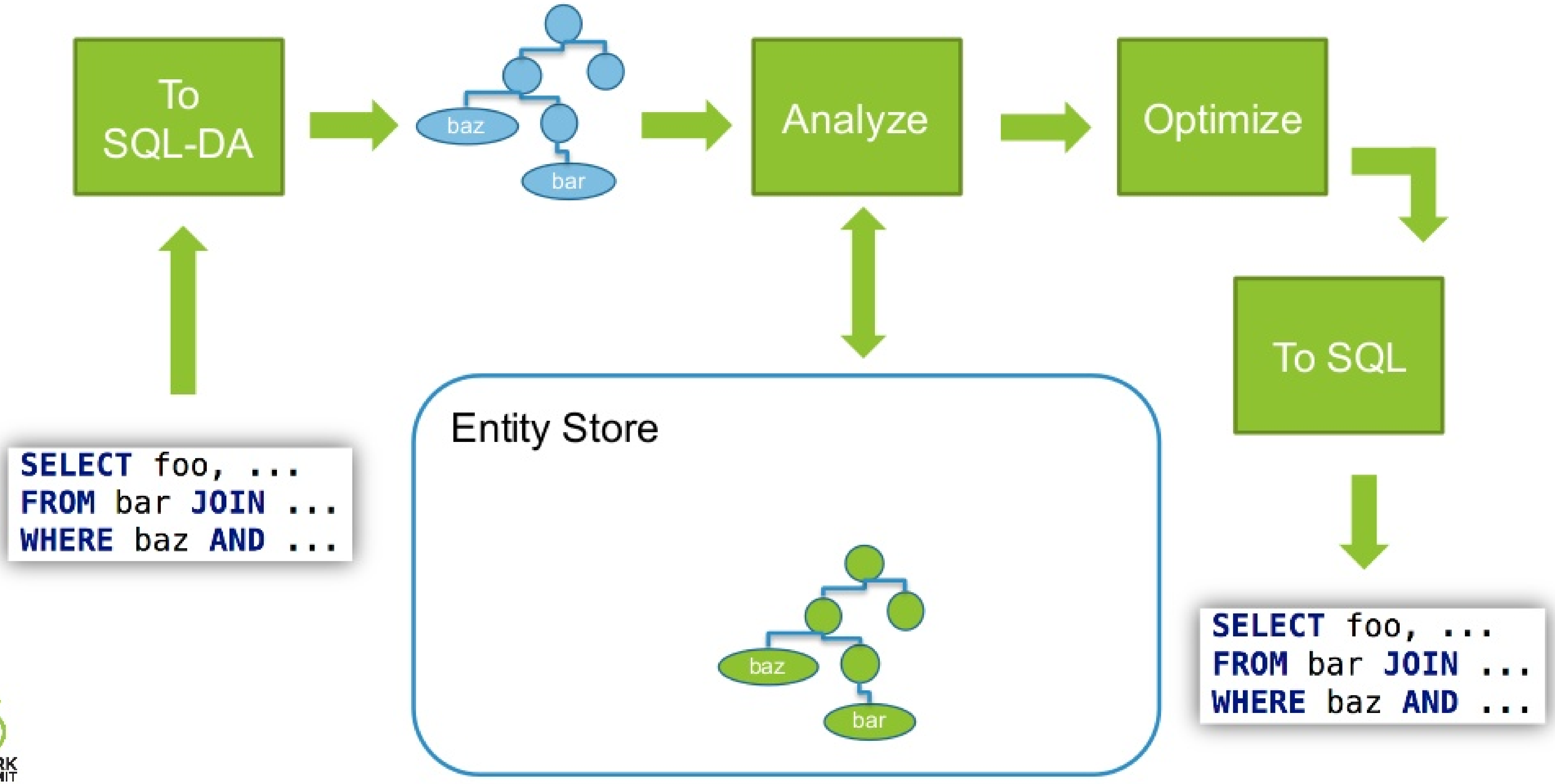


```sql
SELECT *
FROM A JOIN B ON A.a = B.b
WHERE B.b2 < 100
```



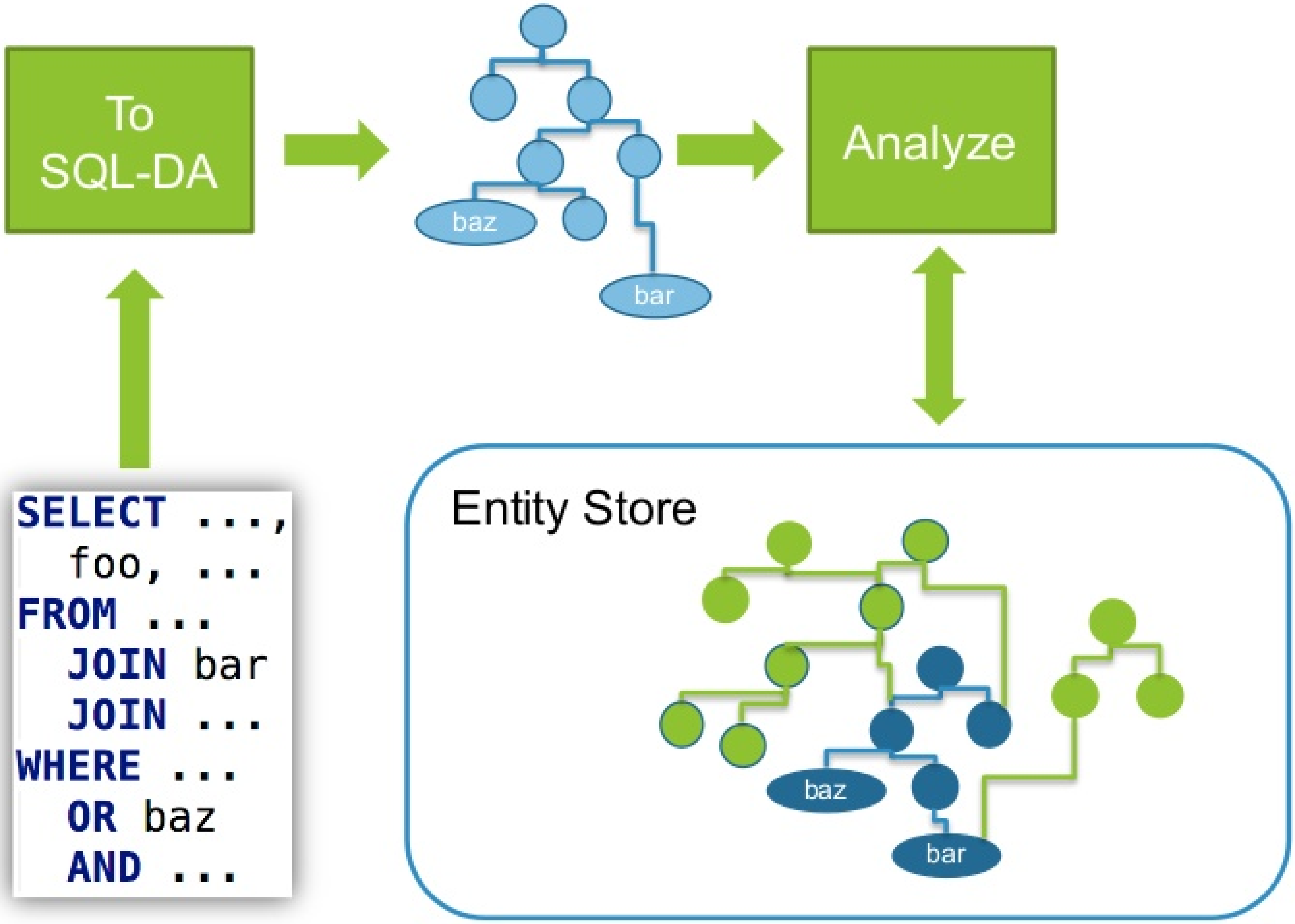
$$A \nabla B = \left(\left\{ \begin{matrix} \{0 \mapsto \alpha, \dots, 3 \mapsto 42.0\}: 3, \\ \{0 \mapsto \beta\}: 1, \\ \dots \end{matrix} \right\}, \begin{pmatrix} 0 & \cdots & 3 & 4 \\ a & \cdots & b & b2 \\ int & \dots & float & int \\ A & A & B & B \end{pmatrix} \right)$$

$$Q := filter_{[4]<100}(filter_{[0]<[3]}(A \nabla B))$$

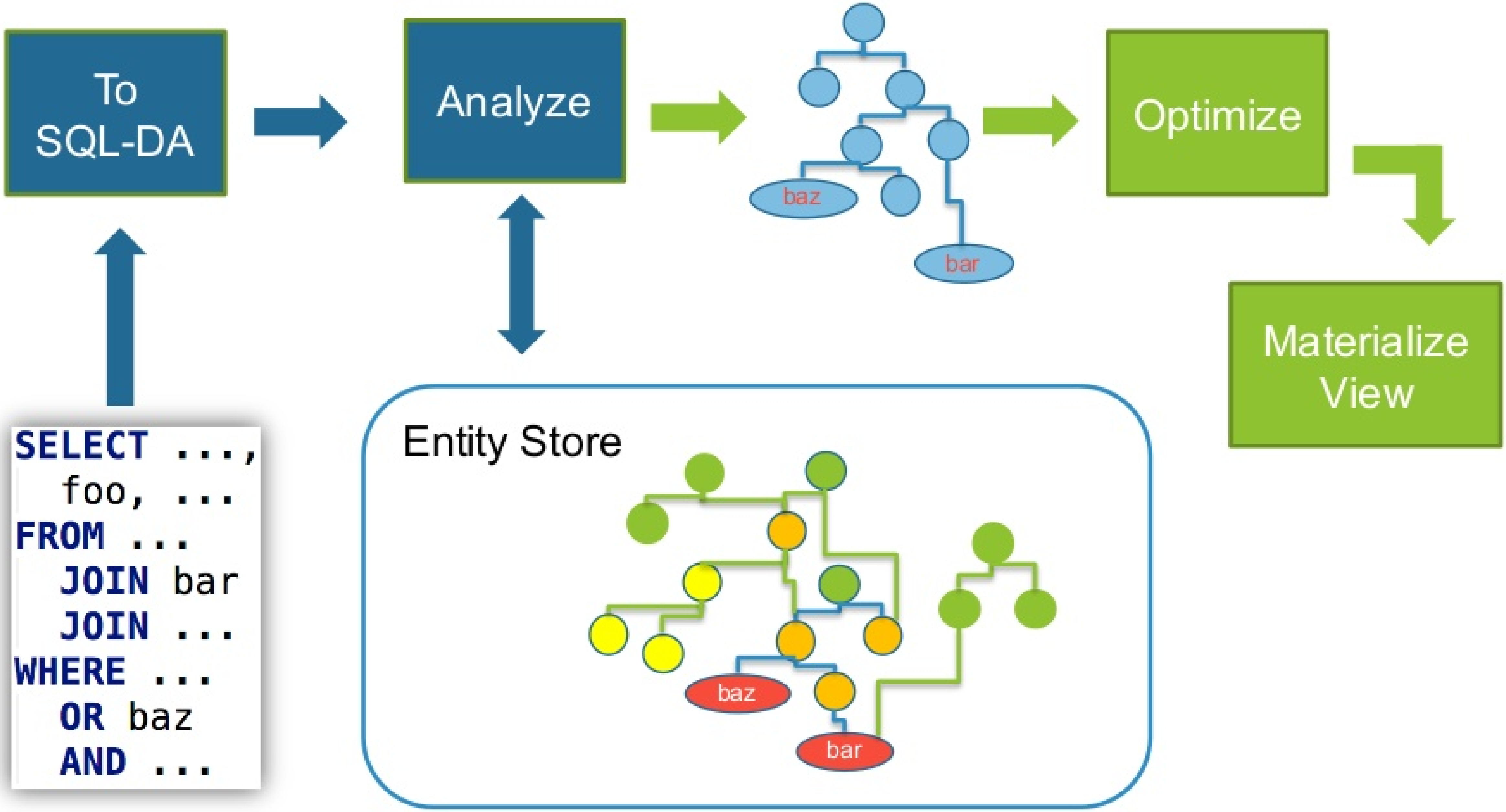


```
SELECT foo, ...
FROM bar JOIN ...
WHERE baz AND ...
```

```
SELECT foo, ...
FROM bar JOIN ...
WHERE baz AND ...
```



```
SELECT ...,
  foo, ...
FROM ...
  JOIN bar
  JOIN ...
WHERE ...
  OR baz
  AND ...
```



```
SELECT ...,
  foo, ...
FROM ...
  JOIN bar
  JOIN ...
WHERE ...
  OR baz
  AND ...
```

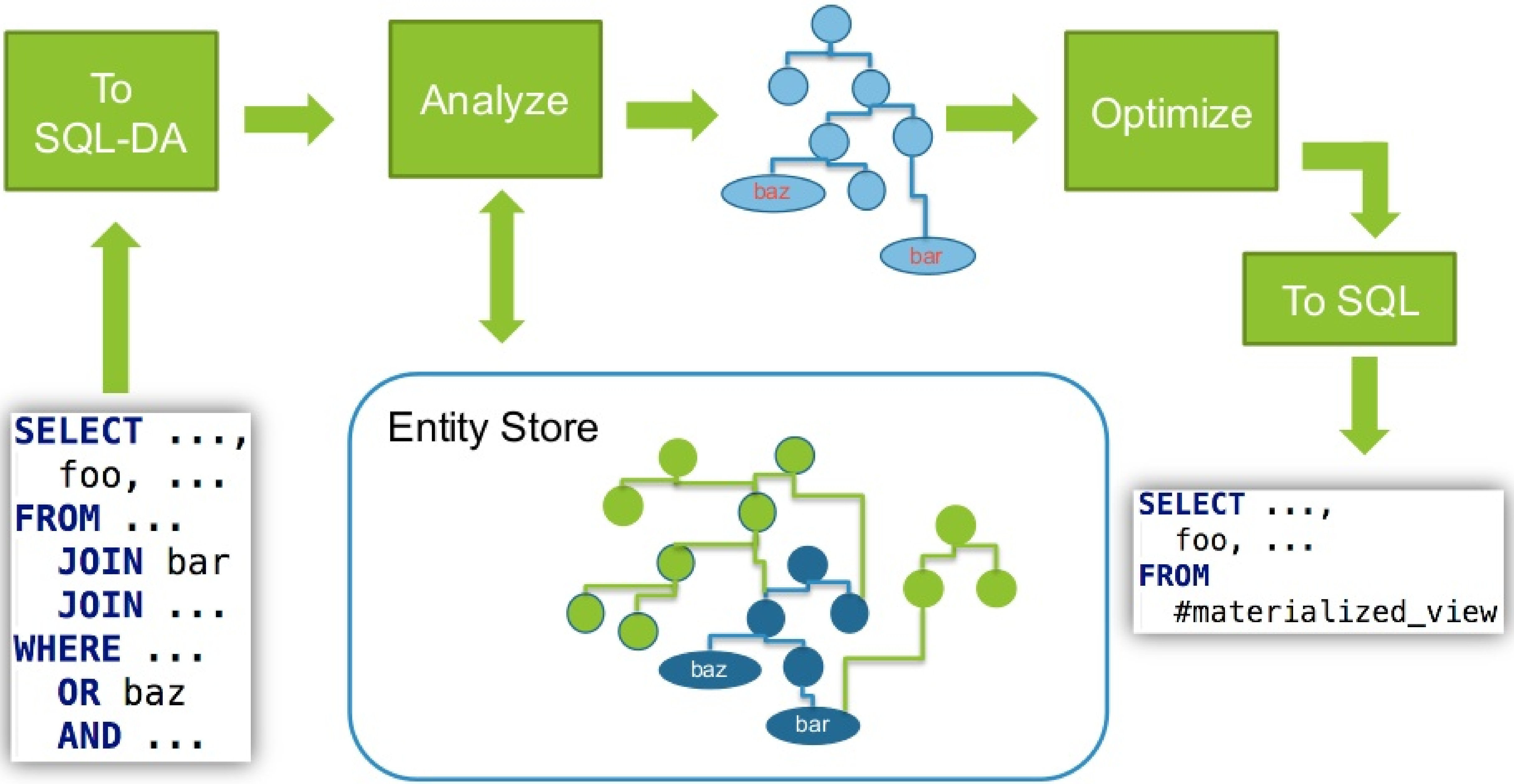
To SQL-DA

Analyze

Optimize

Materialize View

Entity Store



```
SELECT ...,
  foo, ...
FROM ...
  JOIN bar
  JOIN ...
WHERE ...
  OR baz
  AND ...
```

To SQL-DA

Analyze

Optimize

To SQL

Entity Store

```
SELECT ...,
  foo, ...
FROM
  #materialized_view
```

Complex Query Expressions are Turned Into Look-ups

SPARK SUMMIT 2017

Benefits of Autonomous Data Management

- Reduce query time
- Reduce computation resources required
- ***Allow the analyst to focus on problem solving, not data management***

SPARK SUMMIT 2017

Algebraix Inside:

An Implementation of ADM

The PySpark API (DataFrames and SQL) is shimmed.

Before

```
from pyspark import *
from pyspark.sql import SQLContext

conf = SparkConf()
sc = SparkContext(conf=conf)
sqlContext = SQLContext(sc)

names = sc.readText(“people.txt”)

namesDF = sc.createDataFrame(names)

namesDF.registerTempTable(“names”)

sqlContext.sql(”””
    SELECT * FROM names
”””).show()
```

After

```
from aqaspark import *


conf = SparkConf()
sc = SparkContext(conf=conf)
sqlContext = SQLContext(sc)

names = sc.readText(“people.txt”)

namesDF = sc.createDataFrame(names)

namesDF.registerTempTable(“names”)

sqlContext.sql(”””
    SELECT * FROM names
”””).show()
```

Wrap Up

Autonomous Data Management makes Spark great for SQL analytics.

SPARK SUMMIT 2017

Thank You.

algebraix data

@wes_holler

wholler@algebraixdata.com

www.algebraixdata.com

tstraub@algebraixdata.com