Goal-Based Data Production

Spark-Powered Smart Data Warehouse

Sim Simeonov, Founder & CTO, Swoop

petabyte scale data engineering & ML run our business

SPARK SUMMIT 2017

What is a data-powered culture?

answer questions 10x easier with

goal-based data production

simple data production request

top 10 campaigns by gross revenue
running on health sites,
weekly for the past 2 complete weeks,
with cost-per-click and
the click-through rate

ts	rank	campaign	gross_revenue	cpc	pct_ctr
2017-05-08	1	933910738aa9f775441ae69803ebefa0	117810	3.127	0.636
2017-05-08	2	48d408d6d35ec7c34878853c7c09734c	111301	3.081	0.699
2017-05-08	3	700e9d5a56792e1f51cfd4fab8447a1c	106029	2.050	0.521
2017-05-08	4	715489e1198bb79245ff1004569bae6b	89090	3.094	0.304
2017-05-08	5	aba84351f5f95e505fff81666cba0a0c	82878	3.126	0.423
2017-05-08	6	3fa0e50788fb16a7729d587902131d1c	76735	3.079	0.708
2017-05-08	7	4f3bb73cbf7996e29c1bb08893bdb785	64514	1.032	0.530
2017-05-08	8	66a27d0ec5e0c222993c6b9efb5ee588	62736	3.116	0.383
2017-05-08	9	171973923df0416923ab654db1f3c839	62337	1.567	1.543
2017-05-08	10	1cef8619f0cc73884e46f44da559e5e1	57331	1.046	0.682

```
with
origins as (select origin_id, site_vertical from dimension_origins where site_vertical = 'health'),
t1 as (
  select campaign_id, origin_id, clicks, views, billing,
    date(to_utc_timestamp(date_sub(to_utc_timestamp(from_unixtime(uts), 'America/New_York'),
    cast(date_format(to_utc_timestamp(from_unixtime(uts), 'America/New_York'), 'u') as int) - 1), 'America/New_York')) as ts
  from rep_campaigns_daily
  where from_unixtime(uts) >= to_utc_timestamp(date_sub(to_utc_timestamp(current_date(), 'America/New_York'),
        cast(date_format(to_utc_timestamp(current_date(), 'America/New_York'), 'u') as int) - 1) - INTERVAL 2 WEEK,
'America/New_York')
    and from_unixtime(uts) < to_utc_timestamp(date_sub(to_utc_timestamp(current_date(), 'America/New_York'),
        cast(date_format(to_utc_timestamp(current_date(), 'America/New_York'), 'u') as int) - 1), 'America/New_York')
),
t2 as (
  select ts, campaign_id, sum(views) as views, sum(clicks) as clicks, sum(billing) / 1000000.0 as gross_revenue
  from t1 lhs join origins rhs on lhs.origin_id = rhs.origin_id
  group by campaign_id, ts
),
t3 as (select *, rank() over (partition by ts order by gross_revenue desc) as rank from t2),
t4 as (select * from t3 where rank <= 10)
select
  ts, rank, campaign_short_name as campaign,
  bround(gross_revenue) as gross_revenue,
  format_number(if(clicks = 0, 0, gross_revenue / clicks), 3) as cpc,
  format_number(if(views = 0, 0, 100 * clicks / views), 3) as pct_ctr
from t4 lhs join dimension_campaigns rhs on lhs.campaign_id = rhs.campaign_id
order by ts, rank
```

DSL is equally complex

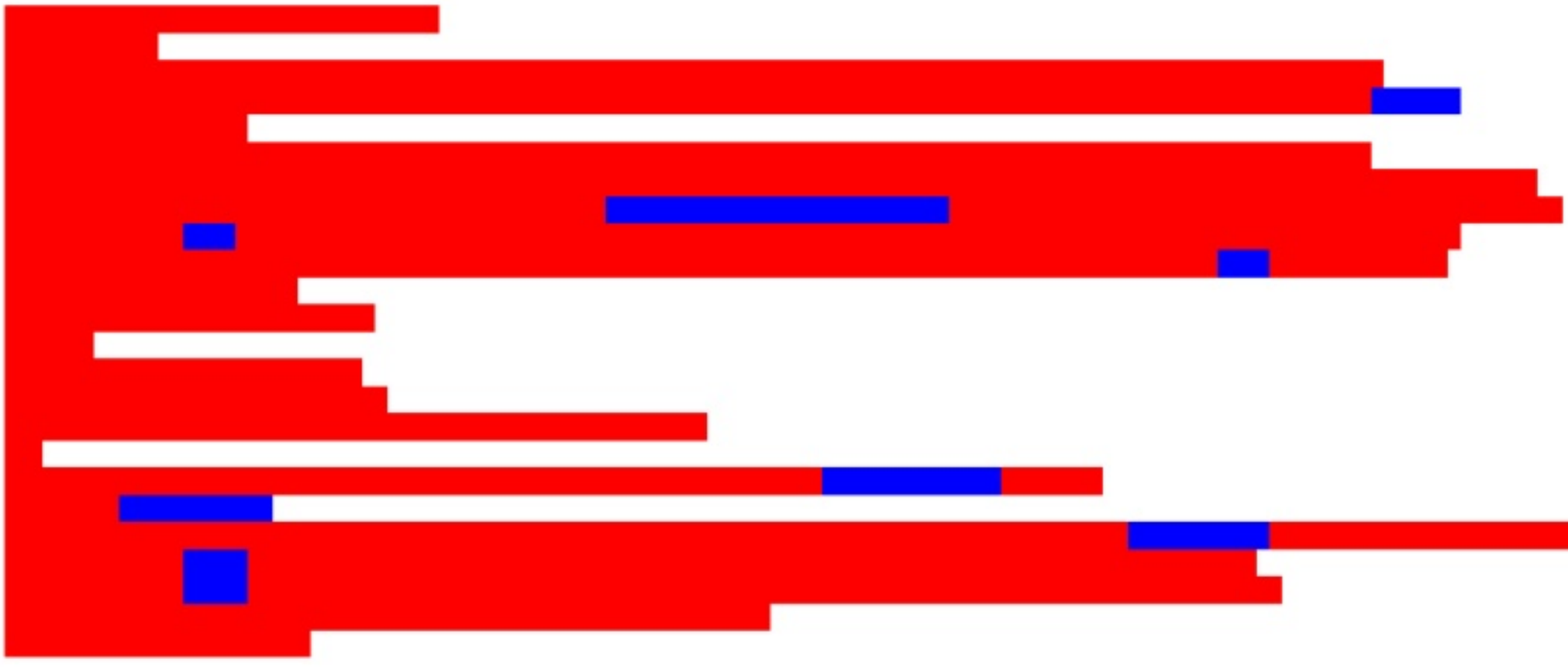
```
spark.table("rep_campaigns_daily")
  .where("""
       from_unixtime(uts)  >= to_utc_timestamp(date_sub(to_utc_timestamp(current_date(),  'America/New_York'),
       cast(date_format(to_utc_timestamp(current_date(),  'America/New_York'),  'u') as int) - 1) - INTERVAL 2 WEEK,  'America/New_York')
    and from_unixtime(uts)  < to_utc_timestamp(date_sub(to_utc_timestamp(current_date(),  'America/New_York'),
       cast(date_format(to_utc_timestamp(current_date(),  'America/New_York'),  'u') as int) - 1), 'America/New_York')""")
  .join(spark.table("dimension_origins").where('site_vertical  === "health").select('origin_id),  "origin_id")
  .withColumn("ts",  expr("date(to_utc_timestamp(date_sub(to_utc_timestamp(from_unixtime(uts),  'America/New_York'),
                    cast(date_format(to_utc_timestamp(from_unixtime(uts),  'America/New_York'),  'u') as int) - 1), 'America/New_York'))"))
  .groupBy('campaign_id,  'ts)
  .agg(
    sum('views).as("views"),
    sum('clicks).as("clicks"),
    (sum('billing)  / 1000000).as("gross_revenue")
  )
  .withColumn("rank",  rank.over(Window.partitionBy('ts).orderBy('gross_revenue.desc)))
  .where('rank  <= 10)
  .join(spark.table("dimension_campaigns").select('campaign_id,  'campaign_short_name.as("campaign")),  "campaign_id")
  .withColumn("gross_revenue",  expr("bround(gross_revenue)"))
  .withColumn("cpc",  format_number(when('clicks  === 0, 0).otherwise('gross_revenue  / 'clicks), 3))
  .withColumn("pct_ctr",  format_number(when('views  === 0, 0).otherwise(lit(100)  * 'clicks / 'views), 3))
  .select('ts,  'rank,  'campaign,  'gross_revenue,  'cpc,  'pct_ctr)
  .orderBy('ts,  'rank)
```

Spark needs to know
what you want and
how to produce it

General data processing requires detailed instructions every single time

the curse of generality: verbosity

what (5%) vs. how (95%)



the curse of generality: duplication

```
-- calculate click-through rate, %
format_number(if(views = 0, 0, 100 * clicks / views), 3) as pct_ctr

-- join to get campaign names from campaign IDs
SELECT campaign_short_name AS campaign, ...
FROM ... lhs
JOIN dimension_campaigns rhs
ON lhs.campaign_id = rhs.campaign_id
```

the curse of generality: complexity

```
-- weekly
date(to_utc_timestamp(
  date_sub(
    to_utc_timestamp(from_unixtime(uts),  'America/New_York'),
      cast(
        date_format(
          to_utc_timestamp(from_unixtime(uts),  'America/New_York'),
          'u')
        as int) - 1),
'America/New_York'))
```

SPARK SUMMIT 2017

the curse of generality: inflexibility

```
-- code depends on time column datatype, format & timezone
date(to_utc_timestamp(
  date_sub(
    to_utc_timestamp(from_unixtime(uts), 'America/New_York'),
      cast(
        date_format(
          to_utc_timestamp(from_unixtime(uts), 'America/New_York'),
          'u')
        as int) - 1),
'America/New_York'))
```

SPARK SUMMIT 2017

Can we keep what and toss how?

Can how become implicit context?

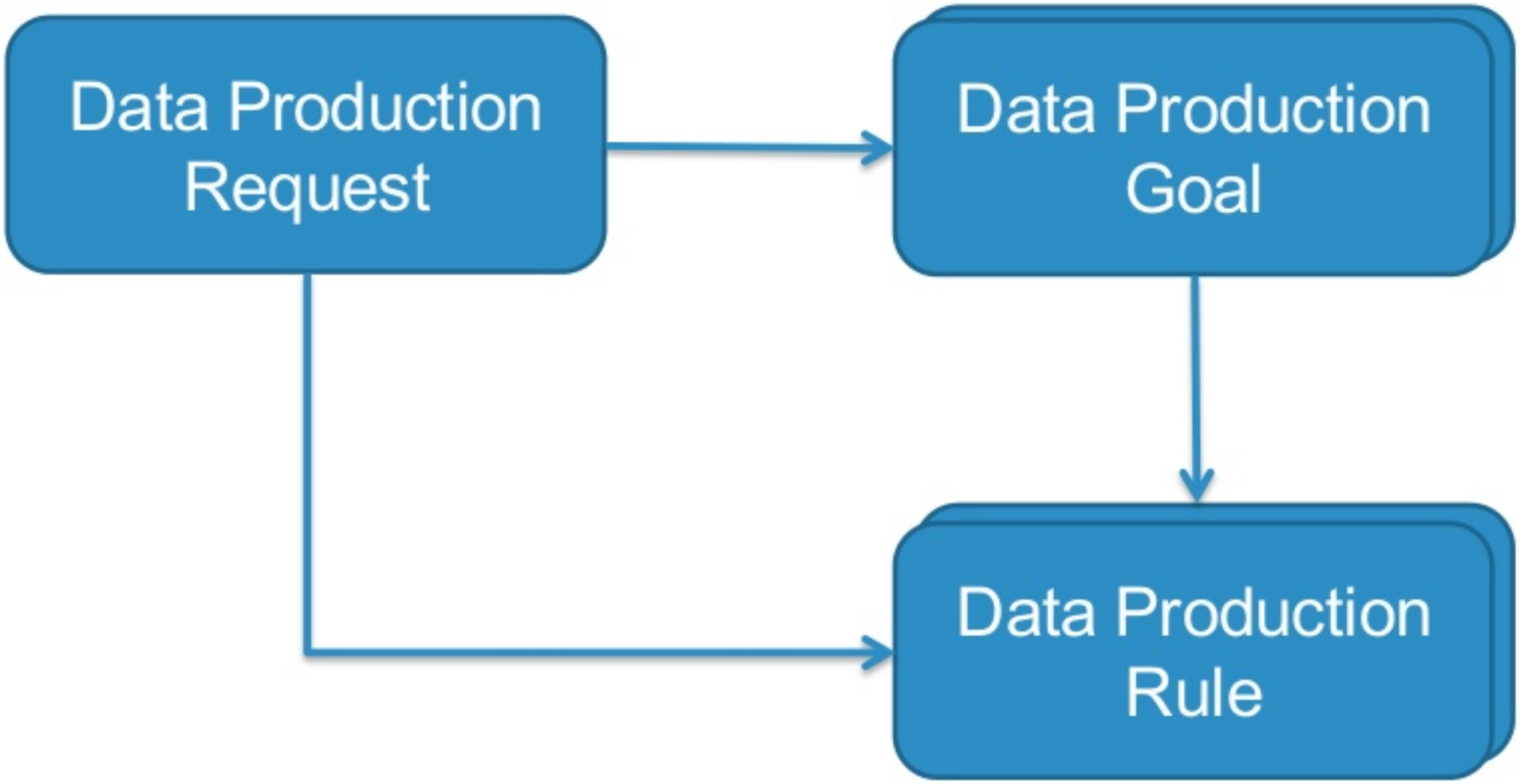
- Data sources, schema, join relationships
- Presentation & formatting
- Week start (Monday)
- Reporting time zone (East Coast)

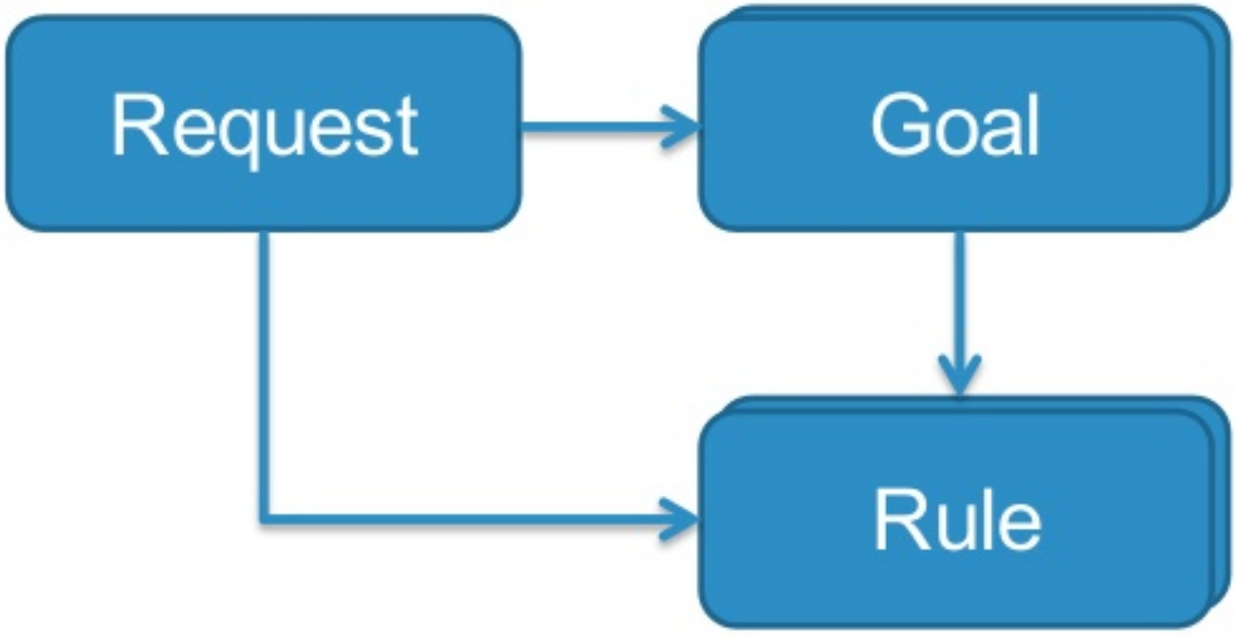
goal-based data production

```
DataProductionRequest()
   .select('campaign.top(10).by('gross_revenue), 'cpc, 'ctr)
   .where("health sites")
   .weekly.time("past 2 complete weeks")
   .humanize
```

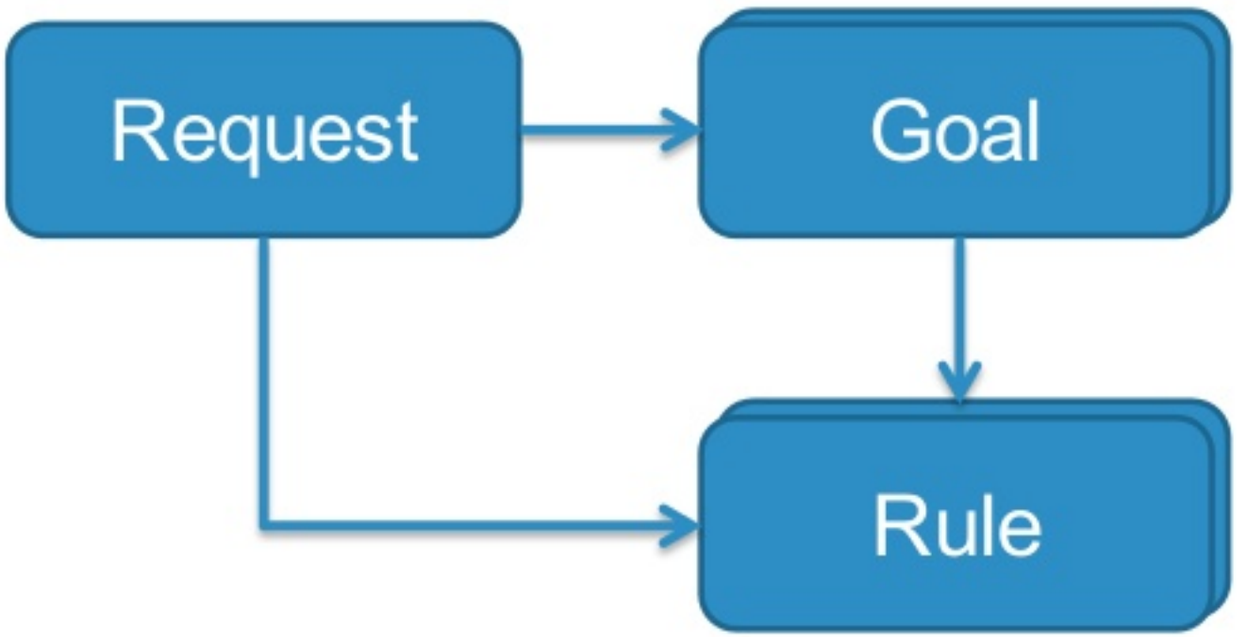
let’s go behind the curtain

the DSL is not as important as the processing & configuration models

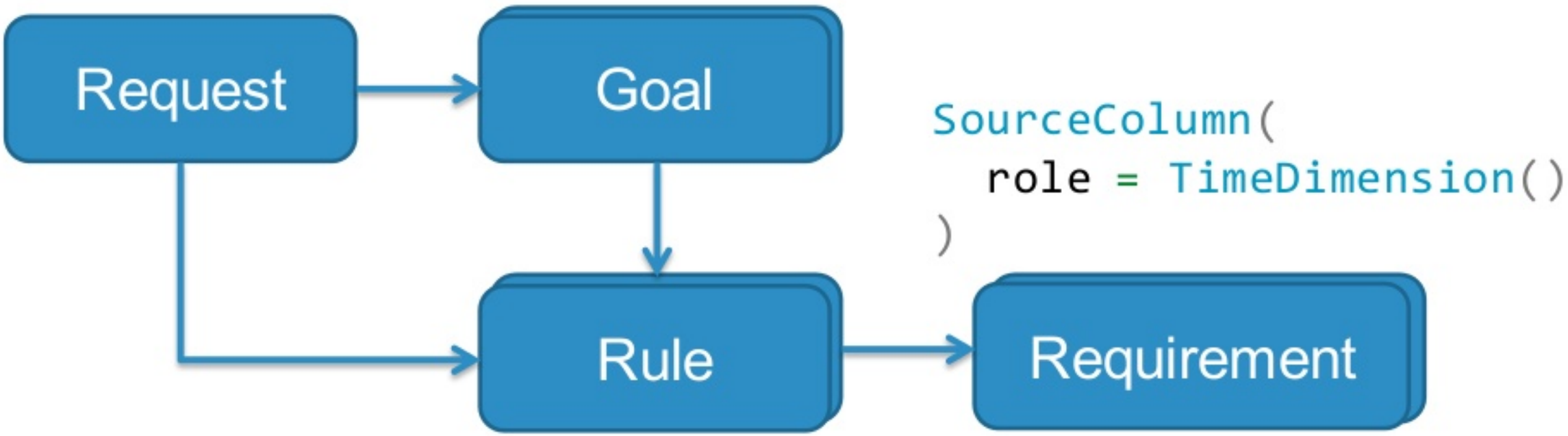

Data Production Request
Data Production Goal
Data Production Rule




```
dpr.addGoal(
  ResultColumn("cpc")
)
```



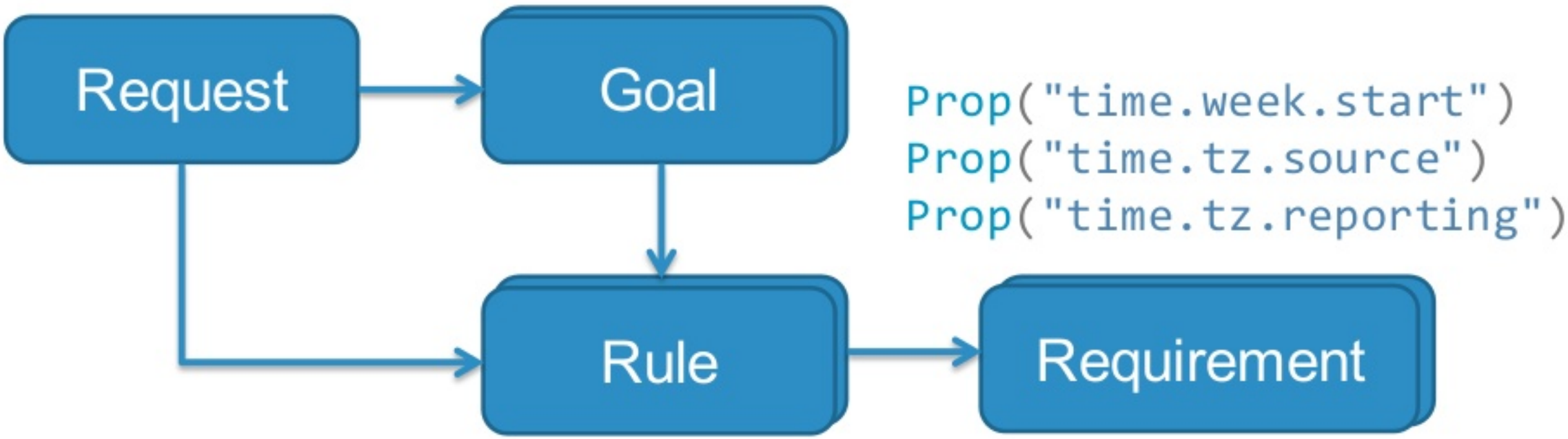

```
// .time("past 2 complete weeks")
dpr.addProductionRule(
  TimeFilter(CompleteWeeks(ref = Now, beg = -2)),
  at = SourceFiltering // production stage
)
```




```
SourceColumn(
   role = TimeDimension()
)
```

```
// .time("past 2 complete weeks")
dpr.addProductionRule(
  TimeFilter(CompleteWeeks(ref = Now, beg = -2)),
  at = SourceFiltering // production stage
)
```

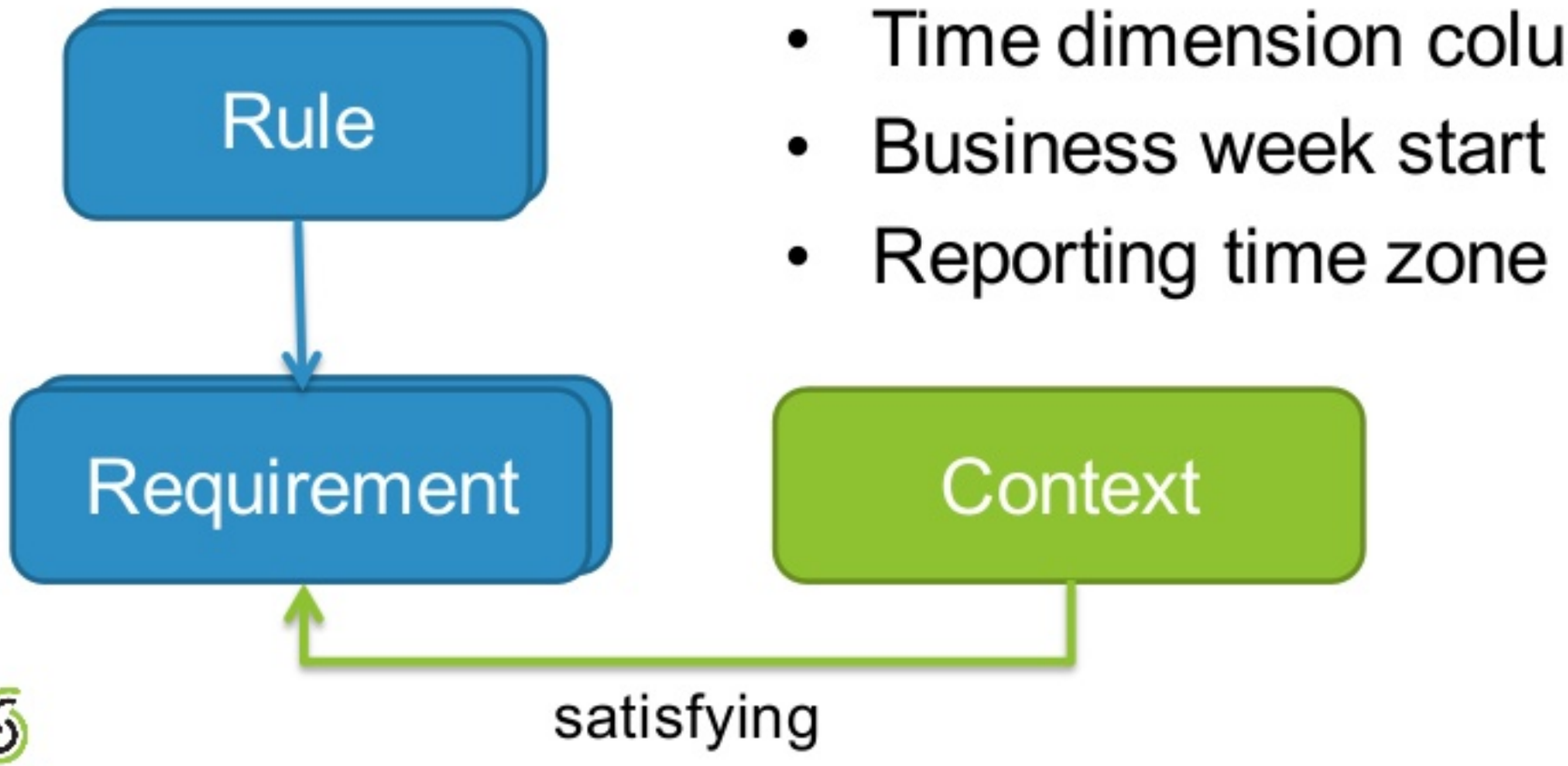
SPARK SUMMIT 2017



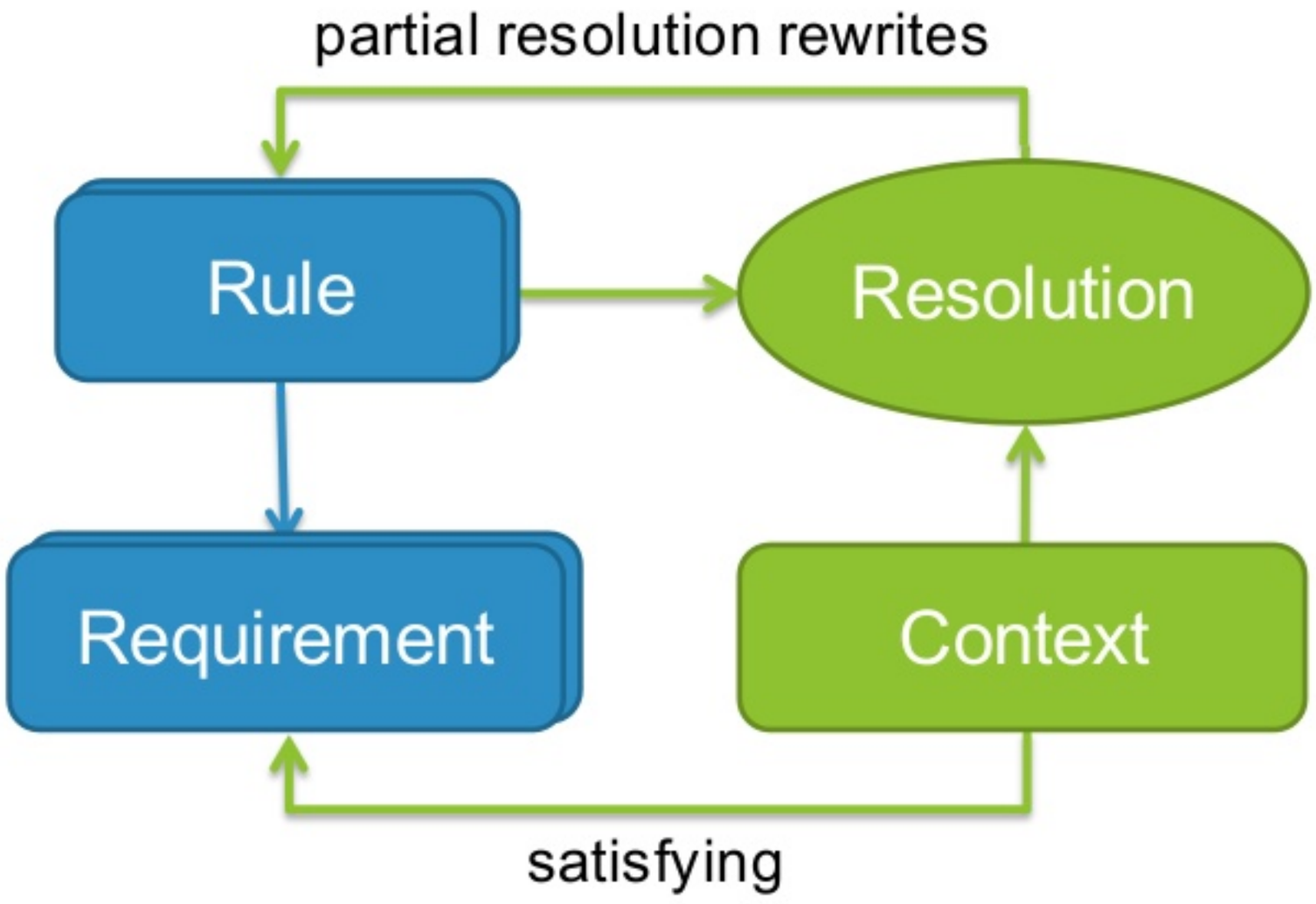

```
// .time("past 2 complete weeks")
dpr.addProductionRule(
  TimeFilter(CompleteWeeks(ref = Now, beg = -2)),
  at = SourceFiltering // production stage
)
```

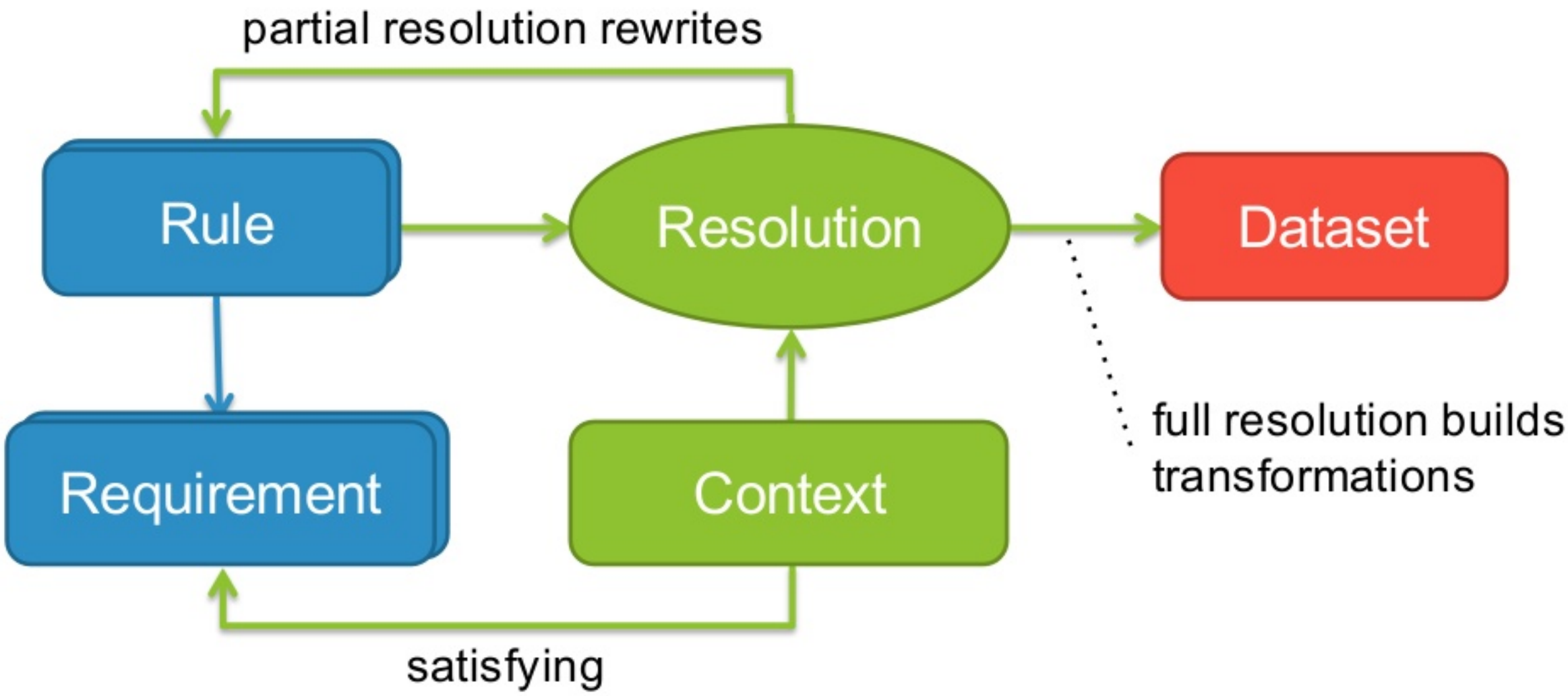
goal + rule + requirement

```
//  .weekly
val timeCol  = SourceColumn(
  role = TimeDimension(maxGranularity  = Weekly)
)
val ts = ResultColumn("ts",
  production  = Array(WithColumn("ts",  BeginningOf("week",  timeCol)))
)
dpr.addGoals(ts).addGroupBy(ts)
```



- Source table
- Time dimension column
- Business week start
- Reporting time zone


partial resolution rewrites
Rule
Resolution
Requirement
Context
satisfying


partial resolution rewrites
Rule
Resolution
Dataset
Requirement
Context
full resolution builds transformations
satisfying

Spark makes this quite easy

- Open & rewritable processing model
- Column = Expression + Metadata
- LogicalPlan

not your typical enterprise/BI death-by-configuration system

context is a metadata query

```
smartDataWarehouse.context
   .release("3.1.2")
   .merge(_.tagged("sim.ml.experiments"))
```

Minimal, just-in-time context is sufficient.

No need for a complete, unified model.

automatic joins by column name

```
{
  "table": "dimension_campaigns",
  "primary_key": ["campaign_id"]
  "joins": [ {"fields": ["campaign_id"]} ]
}
```

join any table with campaign_id column

automatic semantic joins

```
{
  "table": "dimension_campaigns",
  "primary_key": ["campaign_id"]
  "joins": [ {"fields": ["ref:swoop.campaign.id"]} ]
}
```

join any table with a Swoop campaign ID

calculated columns

```
{
  "field": "ctr",
  "description": "click-through rate",
  "expr": "if(nvl(views,0)=0,0,nvl(clicks,0)/nvl(views,0))"
  ...
}
```

automatically available to matching tables

humanization

```
{
  "field": "ctr",
  "humanize": {
    "expr": "format_number(100 * value, 3) as pct_ctr"
    }
}
```

change column name as units change

humanization

```
ResultColumn("rank_campaign_by_gross_revenue",
             production = Array(...),
             shortName = "rank",     // simplified name
             defaultOrdering = Asc, // row ordering
             before = "campaign")   // column ordering
```

optimization hints

```
{
  "field": "campaign",
  // hint: allows join after groupBy as opposed to before
  "unique": true
    ...
}
```

a general optimizer can never do this

automatic source & join selection

- 14+ Swoop tables can satisfy the request
- Only 2 are good choices based on cost
  - 100+x faster execution

10x easier data production

```
val df = DataProductionRequest()
  .select('campaign.top(10).by('gross_revenue), 'cpc, 'ctr)
  .where("health sites")
  .weekly.time("past 2 complete weeks")
  .humanize
  .toDF // result is a normal dataframe/dataset
```

revolutionary benefits

- Increased productivity & flexibility
- Improved performance & lower costs
- Easy collaboration (even across companies!)
- Business users can query Spark

At Swoop, we are committed to making
goal-based data production
the best free & open-source
interface for Spark data production

Interested in contributing?
Email spark@swoop.com

SPARK SUMMIT 2017

Join our open-source work.

Email spark@swoop.com

More Spark magic: http://bit.ly/spark-records