Apache Kylin 2.0

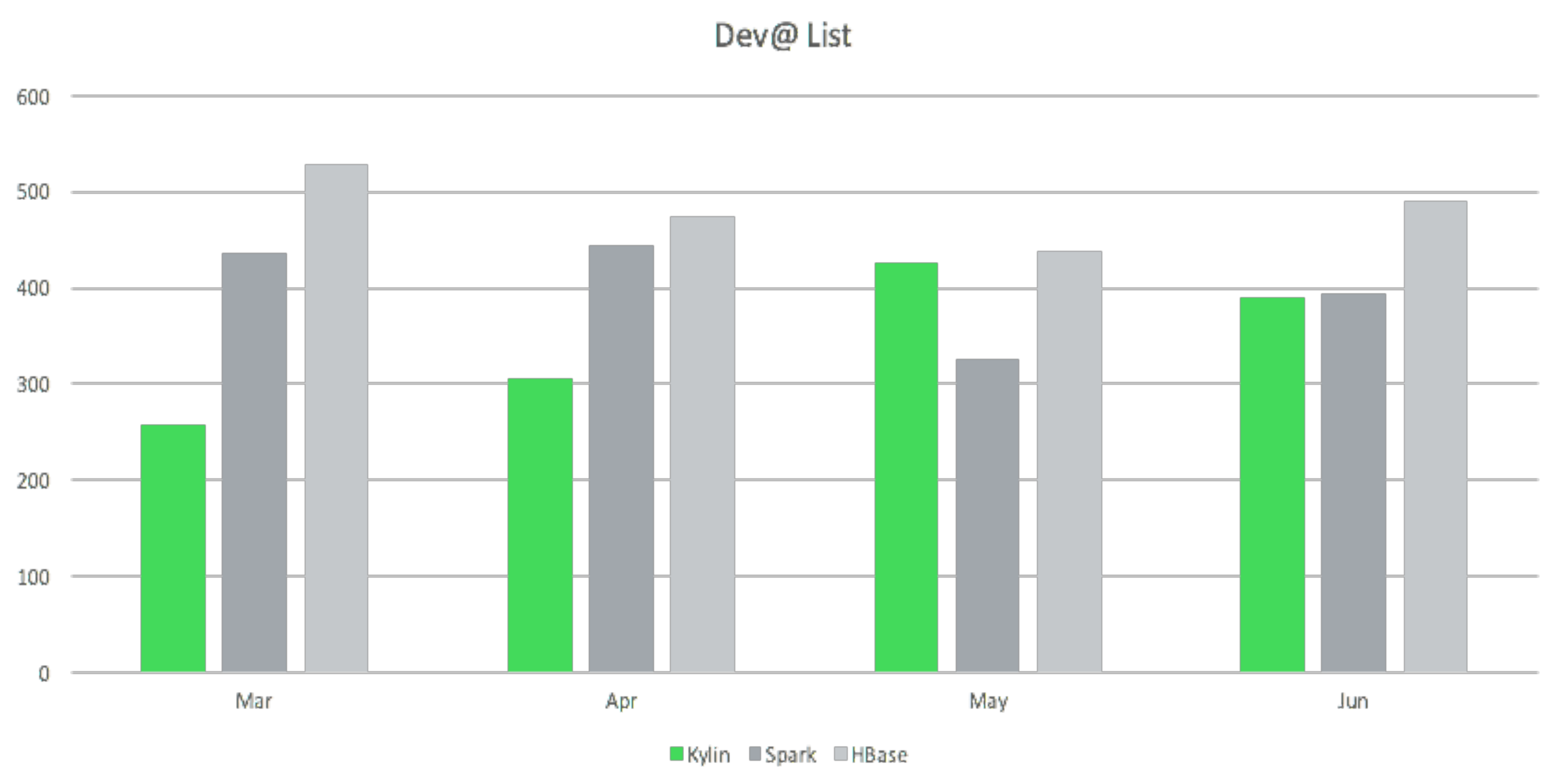
从传统OLAP到实时数据仓库

李 栋

关于我

- **李栋**
  - Apache Kylin Committer & PMC Member
  - Kyligence Inc. 技术合伙人兼高级软件架构师
  - kybot.io技术负责人

Apache Kylin
超高性能的OLAP on Hadoop大数据分析引擎



开源社区活跃度

✓**首个来自中国的Apache顶级开源项目**

✓核心团队都是中国人

✓连续两年荣获InfoWorld Bossie Award“**最佳开源大数据工具奖**”，同时获奖的还有Google TensorFlow、Apache Spark等

Apache Kylin发展历程

InfoWorld: Bossie Award

最佳开源大数据工具奖

(2015 & 2016)

项目开始

Sept 2013

正式开源

Oct 2014

加入Apache
孵化器项目

Nov 2014

Sept 2015

毕业成为Apache
顶级项目

Nov 2015

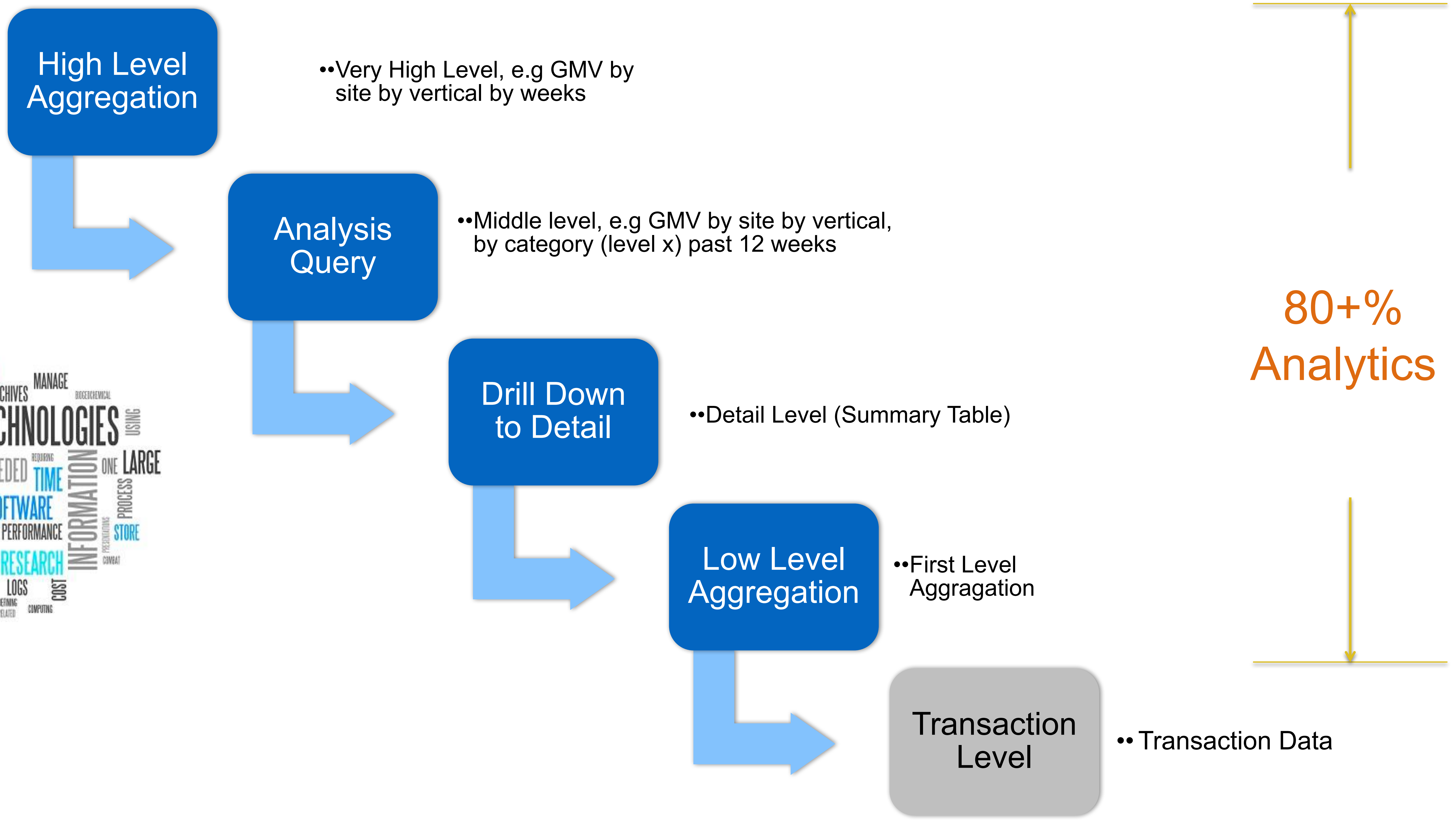
Kyligence公司创建

Mar 2016

什么是Apache Kylin？

OSC源创会 Opensource Innovation Meetup | IT大咖说 不止于技术

OLAP

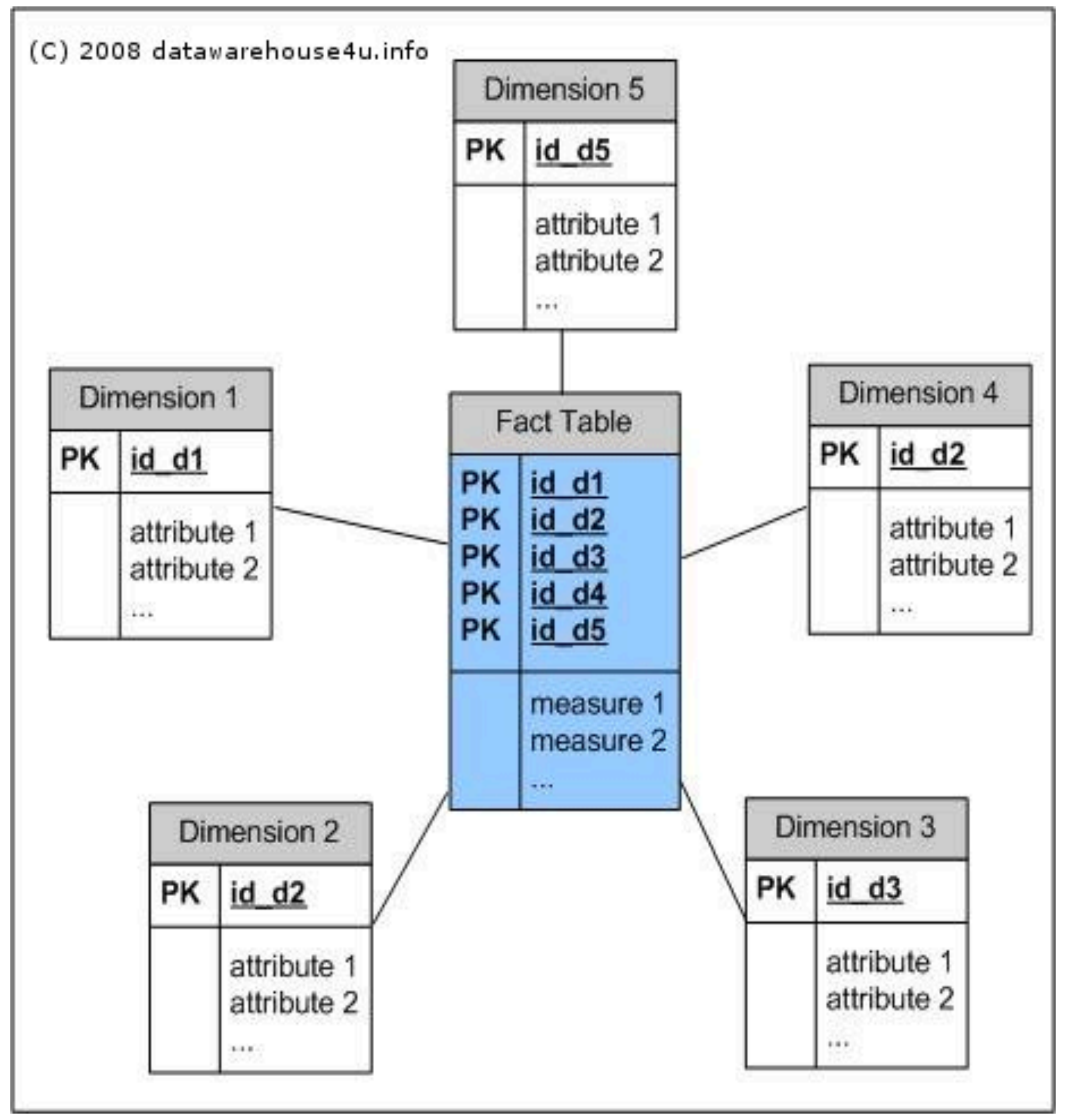


High Level Aggregation
••Very High Level, e.g GMV by site by vertical by weeks
Analysis Query
••Middle level, e.g GMV by site by vertical, by category (level x) past 12 weeks
Drill Down to Detail
••Detail Level (Summary Table)
Low Level Aggregation
••First Level Aggragation
Transaction Level
•• Transaction Data
80+% Analytics

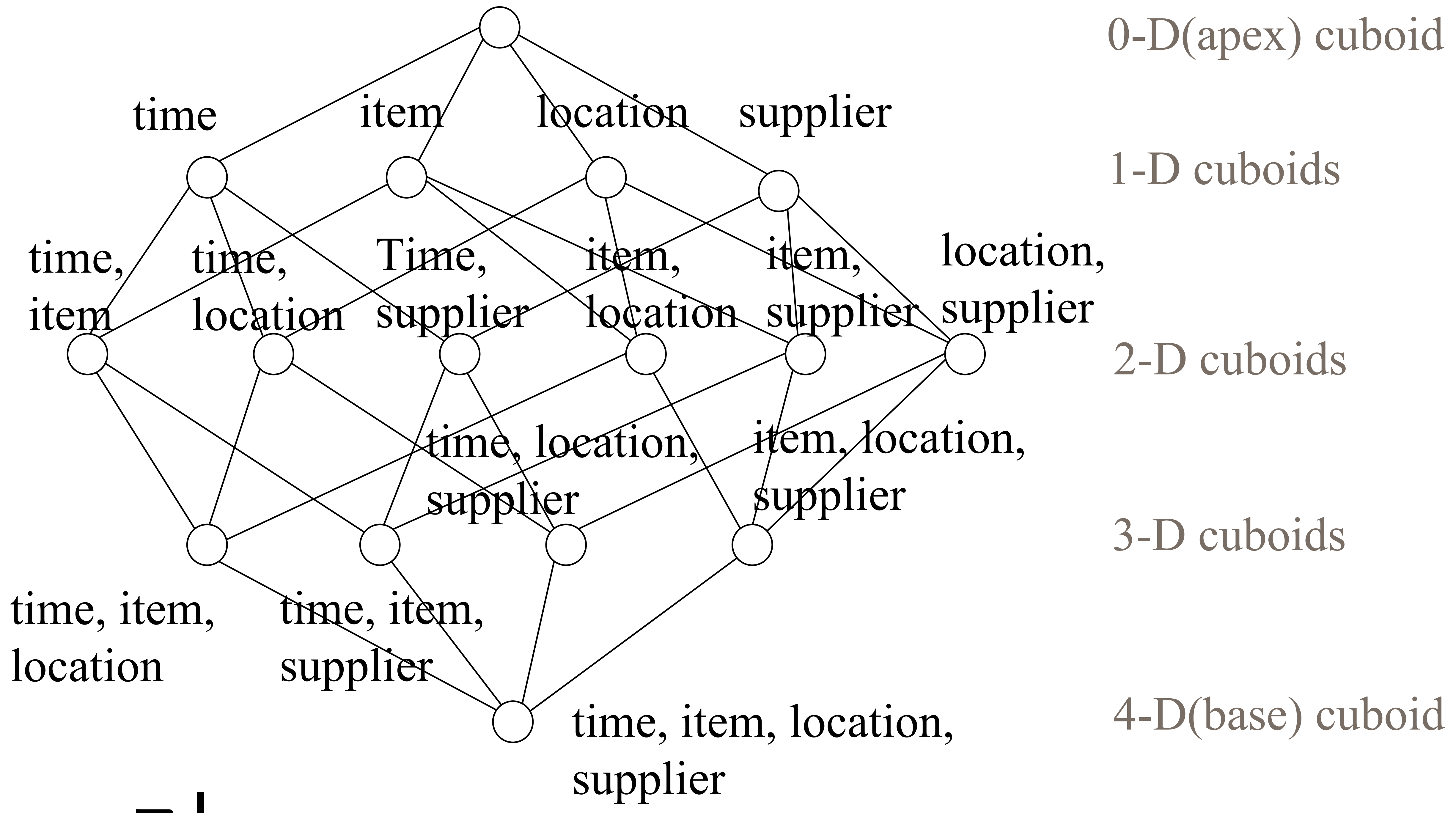
Kylin is designed to accelerate analytics queries performance on Hadoop.

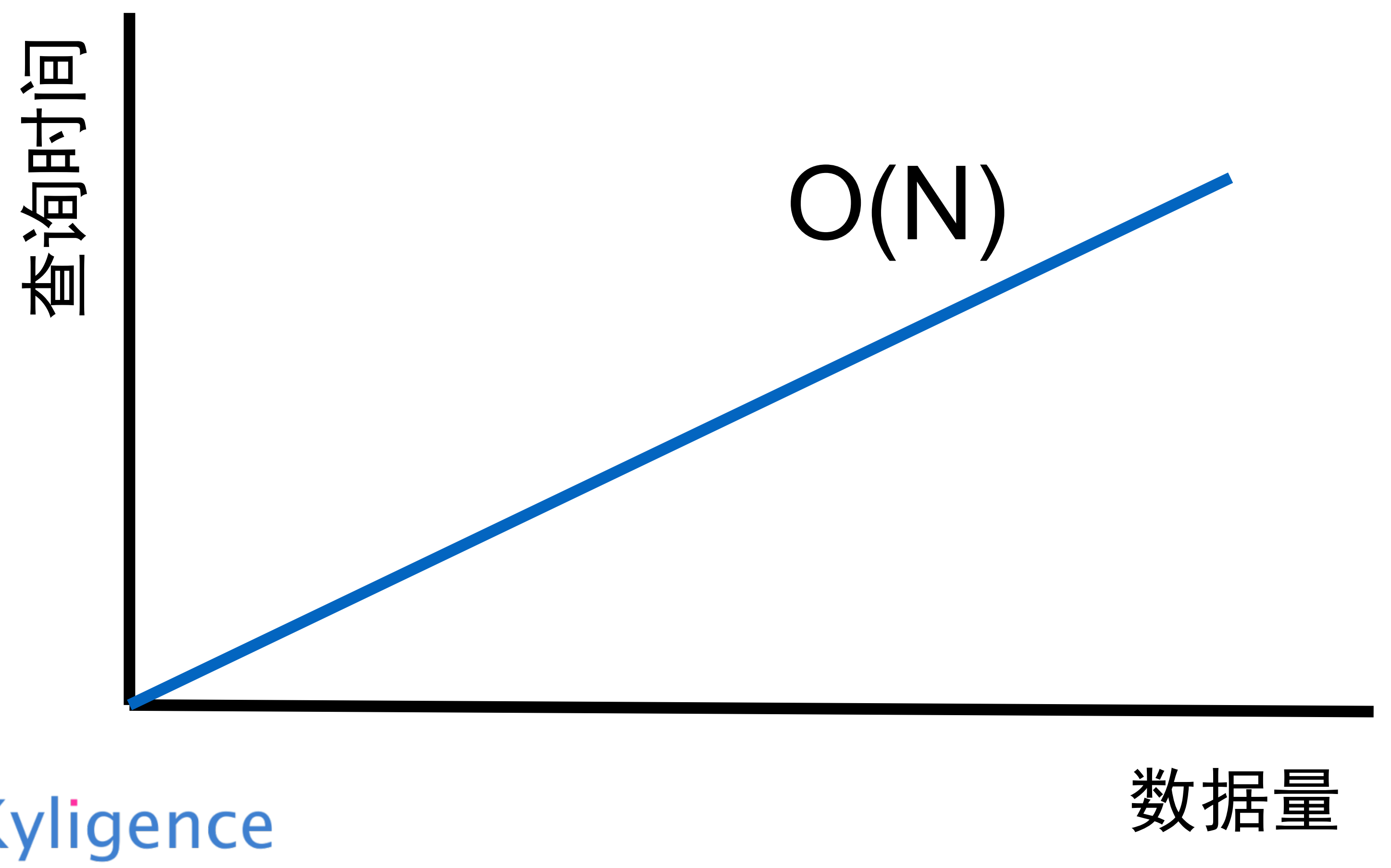
Kyligence

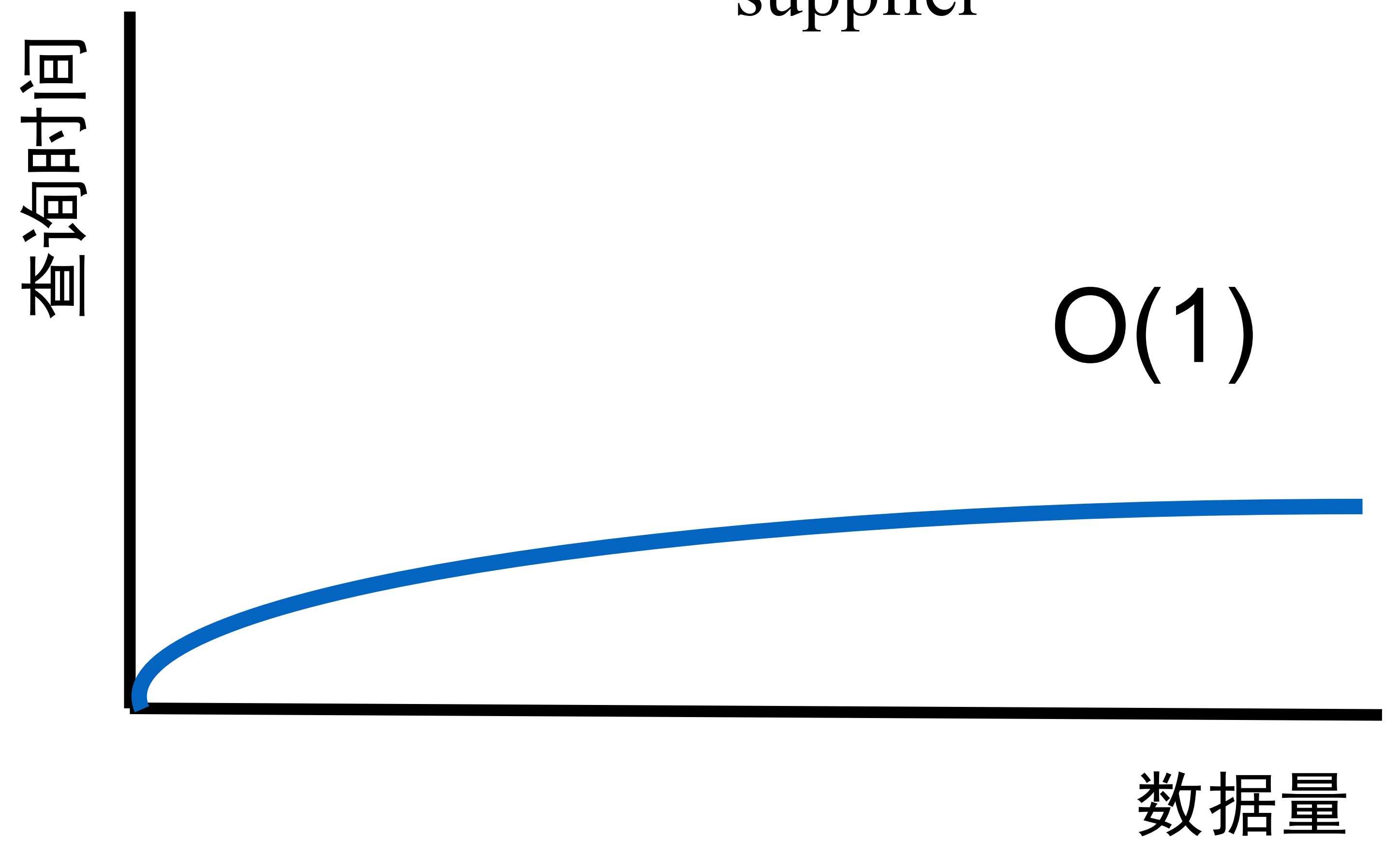
原理：空间换时间（Cube）



预计算

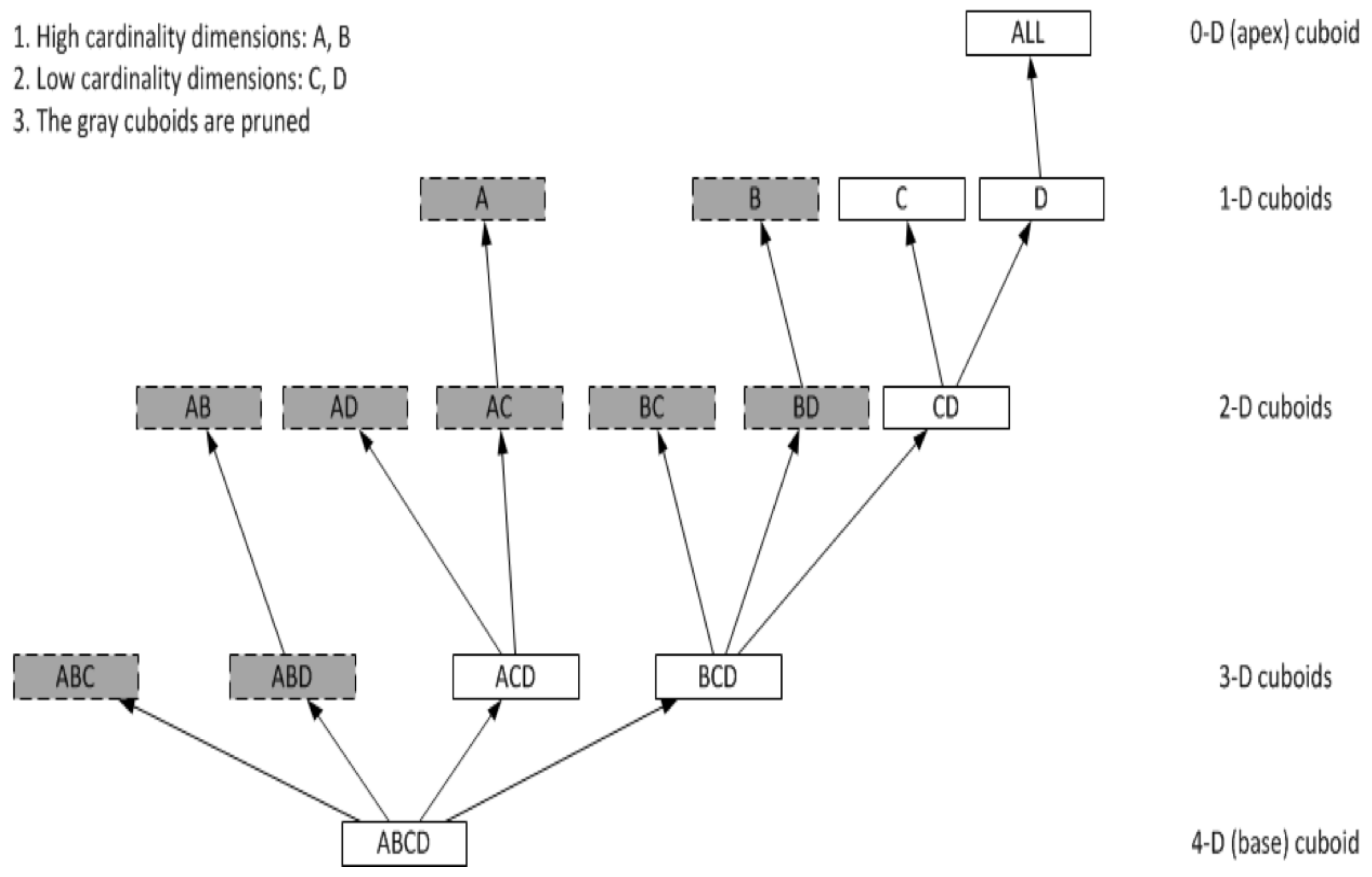






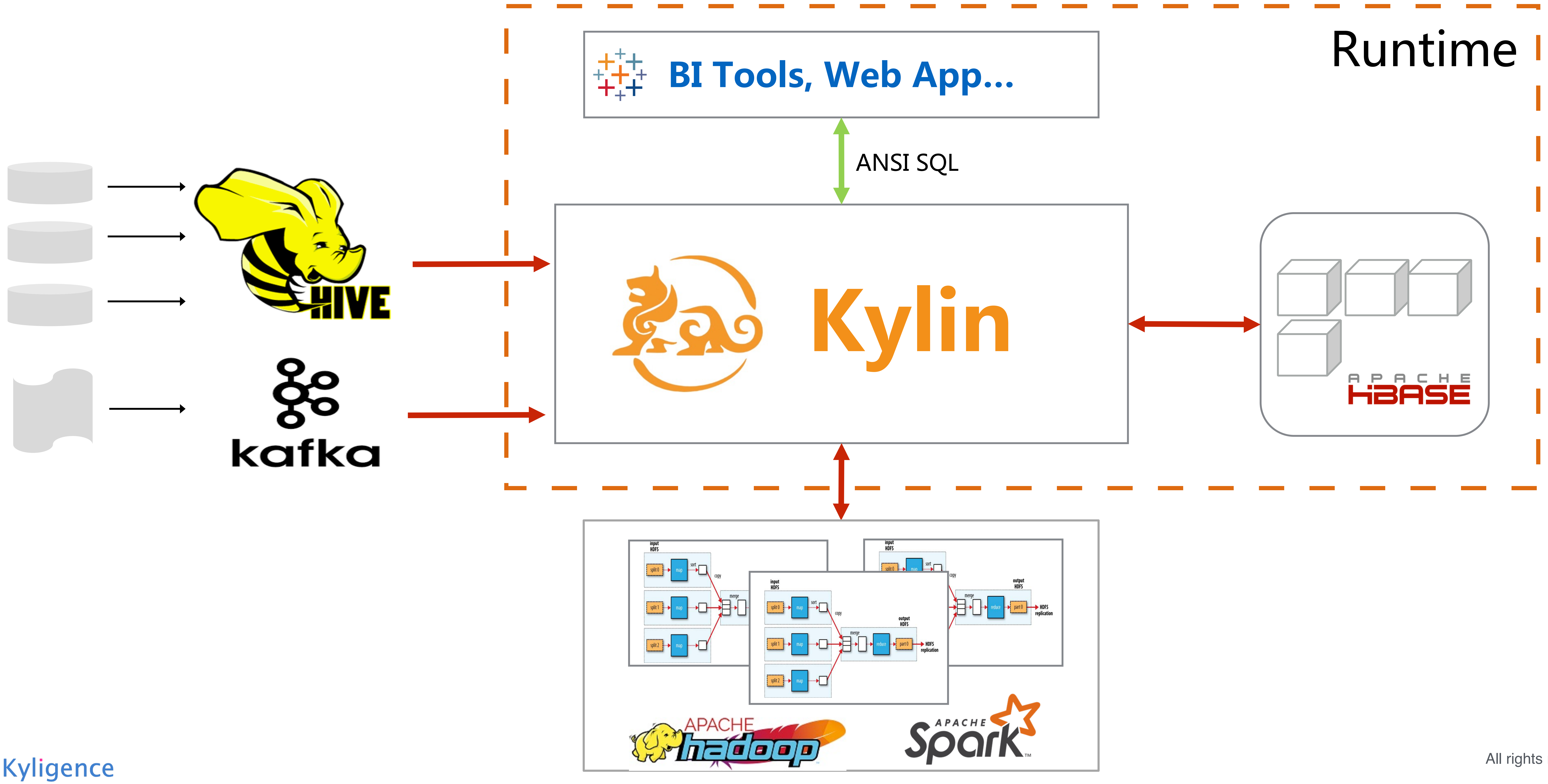
如何解决数据爆炸问题？

- **降维**
  - 衍生 (Derived)
  - 必须 (Mandatory)
  - 层级 (Hierarchy)
  - 联合 (Joint)
- **维度分组**
  - $2^{30} \rightarrow 2^{10} + 2^{10} + 2^{10}$



Partial Cube

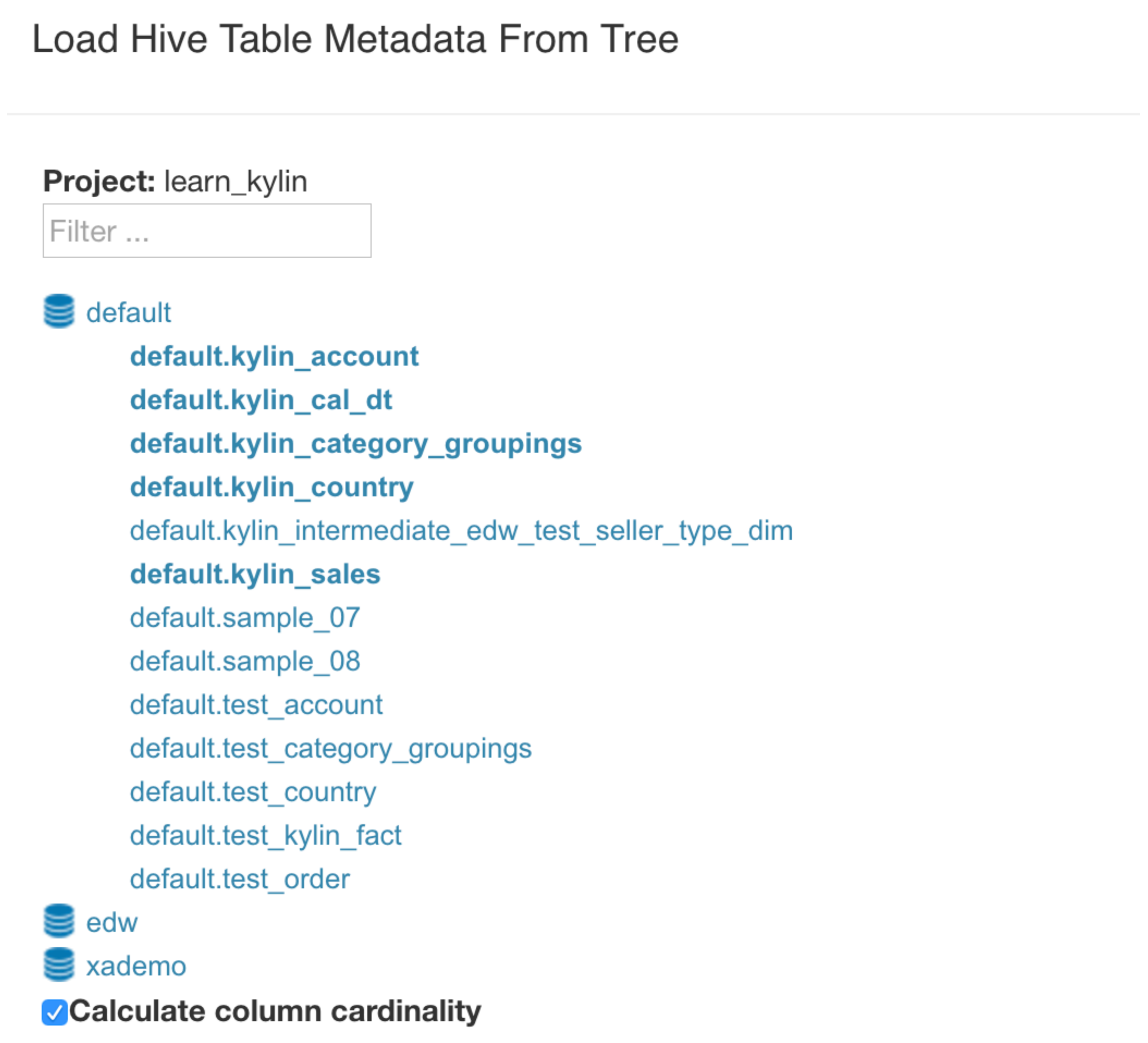
技术架构



Kyligence

创建数据模型

1. 选择Hive表作为数据源


Load Hive Table Metadata From Tree

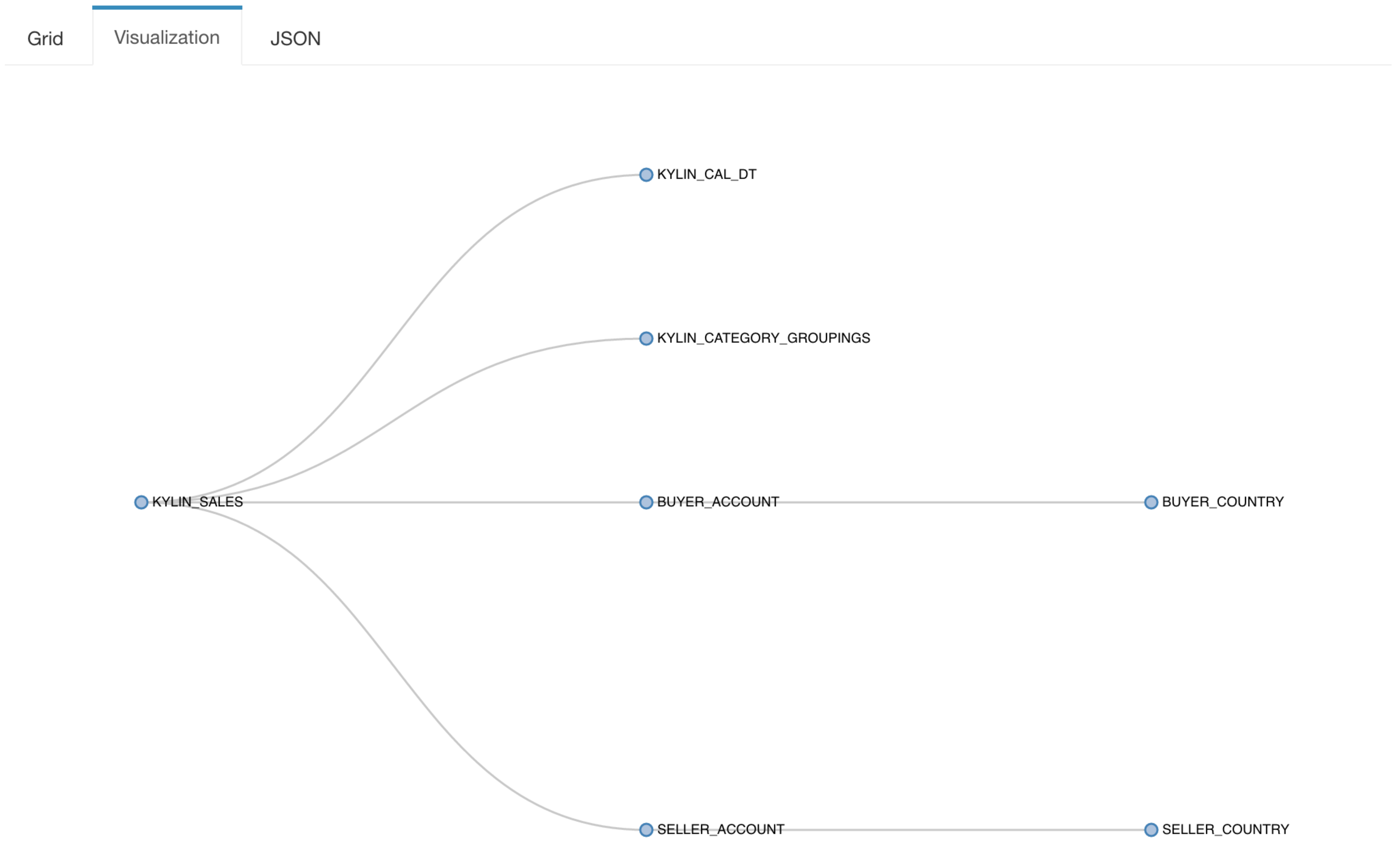
Project: learn_kylin

Filter ...

- default
  - **default.kylin_account**
  - **default.kylin_cal_dt**
  - **default.kylin_category_groupings**
  - **default.kylin_country**
  - default.kylin_intermediate_edw_test_seller_type_dim
  - **default.kylin_sales**
  - default.sample_07
  - default.sample_08
  - default.test_account
  - default.test_category_groupings
  - default.test_country
  - default.test_kylin_fact
  - default.test_order
- edw
- xademo

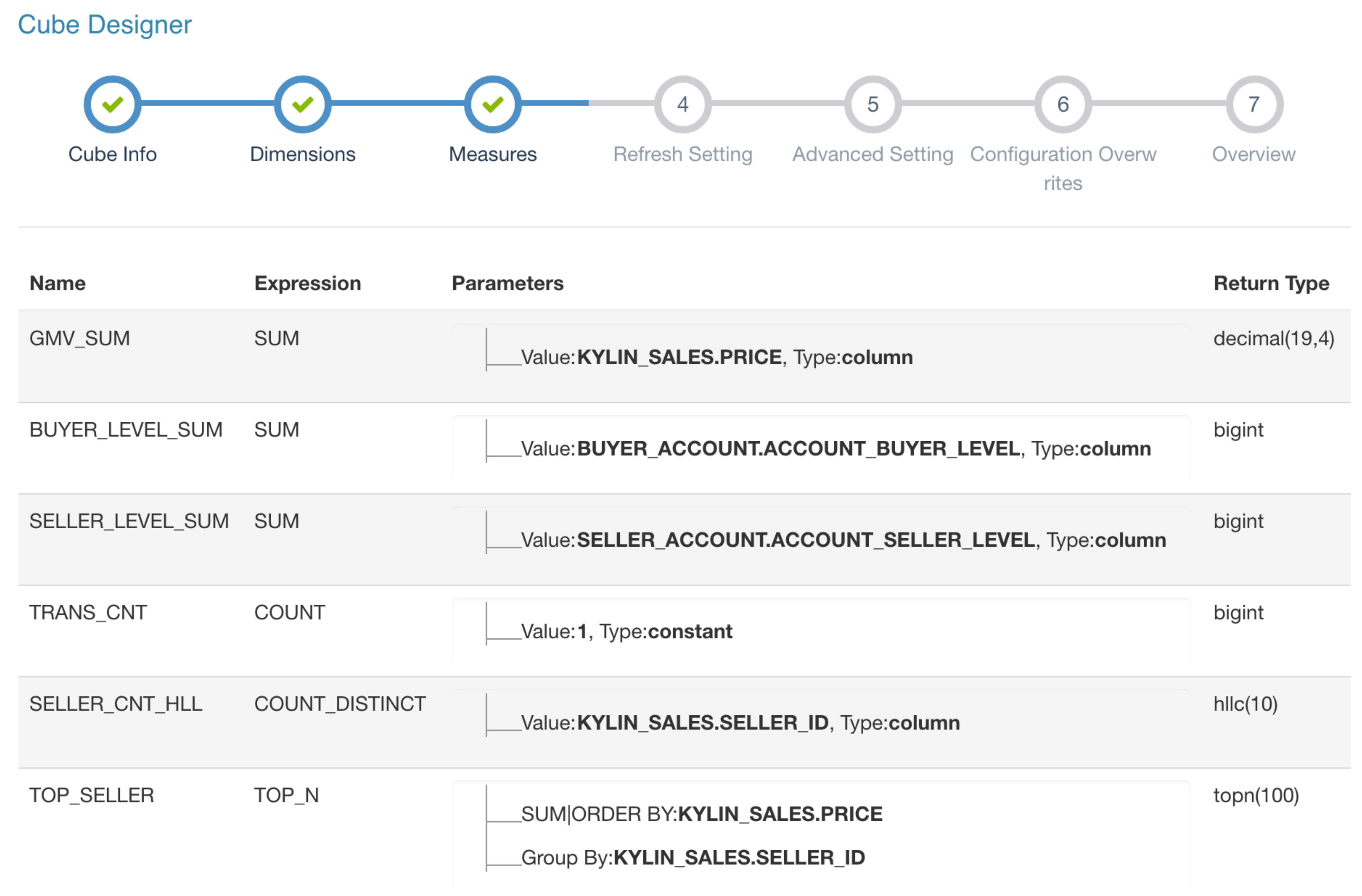
✓ **Calculate column cardinality**

2. 根据ER模型创建数据模型



设计和构建Cube

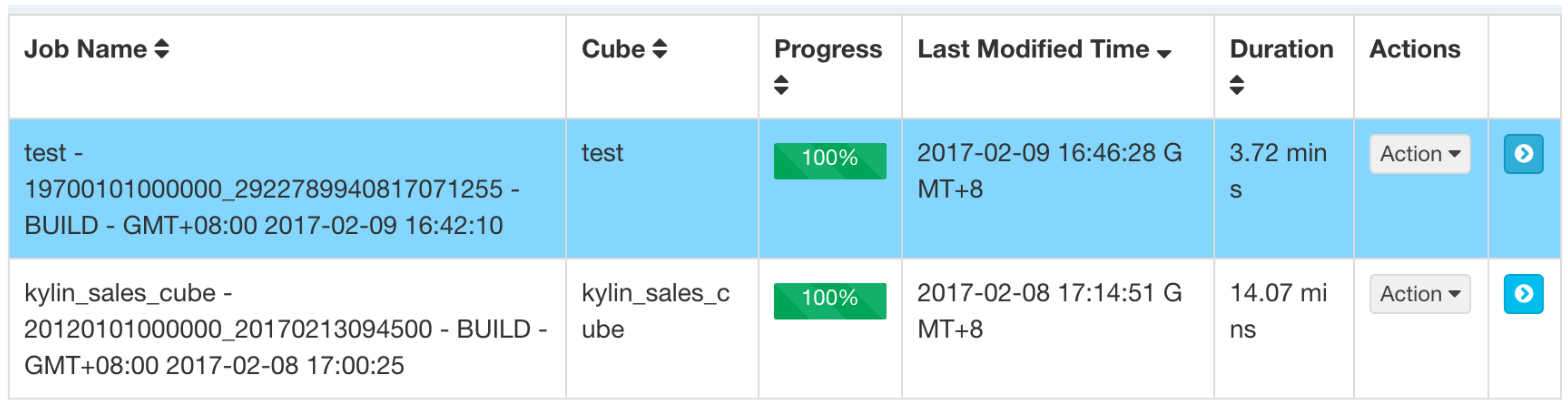
3. 新建Cube（预计算定义）



Cube Designer

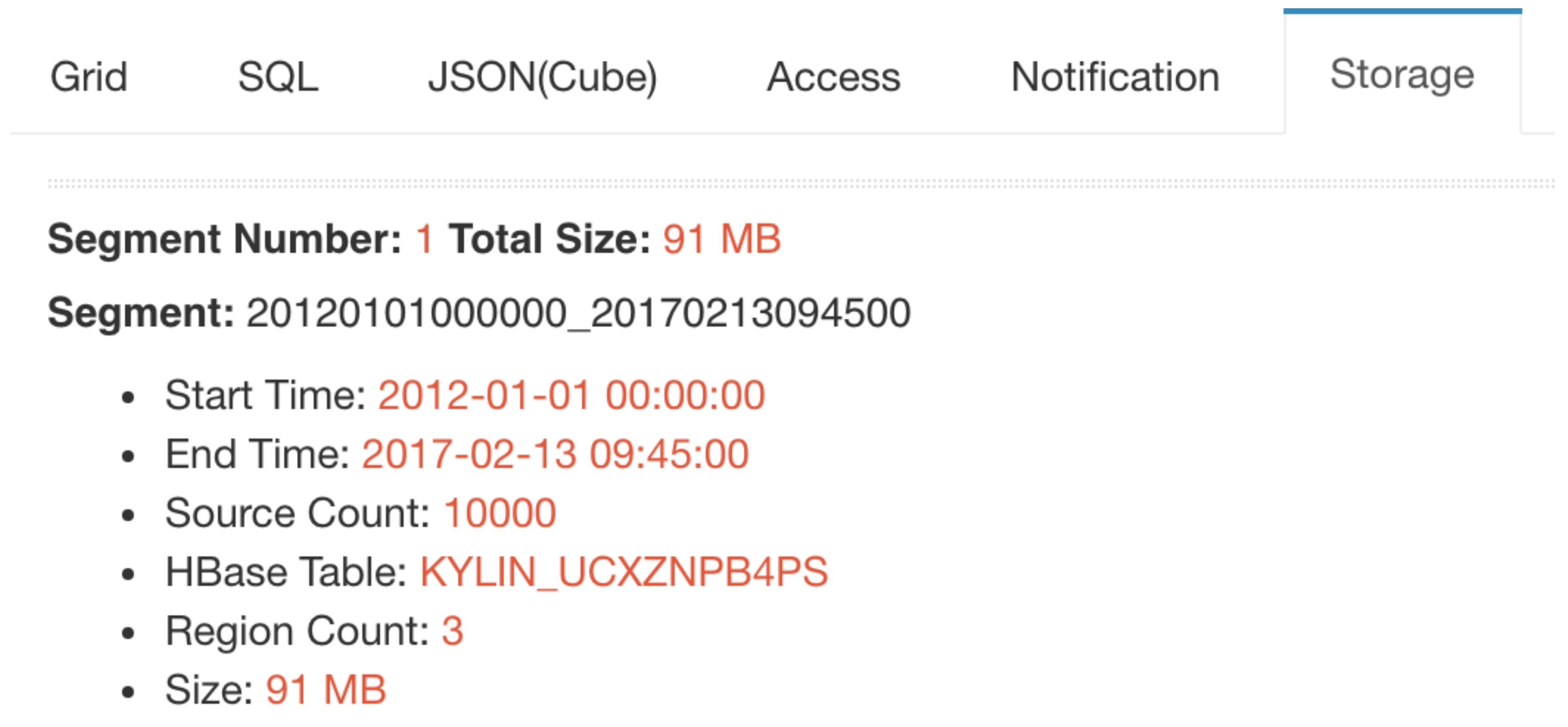
Name	Expression	Parameters	Return Type
GMV_SUM	SUM	Value:**KYLIN_SALES.PRICE**, Type:**column**	decimal(19,4)
BUYER_LEVEL_SUM	SUM	Value:**BUYER_ACCOUNT.ACCOUNT_BUYER_LEVEL**, Type:**column**	bigint
SELLER_LEVEL_SUM	SUM	Value:**SELLER_ACCOUNT.ACCOUNT_SELLER_LEVEL**, Type:**column**	bigint
TRANS_CNT	COUNT	Value:**1**, Type:**constant**	bigint
SELLER_CNT_HLL	COUNT_DISTINCT	Value:**KYLIN_SALES.SELLER_ID**, Type:**column**	hllc(10)
TOP_SELLER	TOP_N	SUM|ORDER BY:**KYLIN_SALES.PRICE** Group By:**KYLIN_SALES.SELLER_ID**	topn(100)

4. 构建Cube（预计算执行）



Job Name ⇕	Cube ⇕	Progress ⇕	Last Modified Time ▾	Duration ⇕	Actions	
test - 19700101000000_2922789940817071255 - BUILD - GMT+08:00 2017-02-09 16:42:10	test	100%	2017-02-09 16:46:28 GMT+8	3.72 mins	Action ▾	⊙
kylin_sales_cube - 20120101000000_20170213094500 - BUILD - GMT+08:00 2017-02-08 17:00:25	kylin_sales_cube	100%	2017-02-08 17:14:51 GMT+8	14.07 mins	Action ▾	⊙

5. 构建完成，可以查询了！



Segment Number: 1 **Total Size:** 91 MB

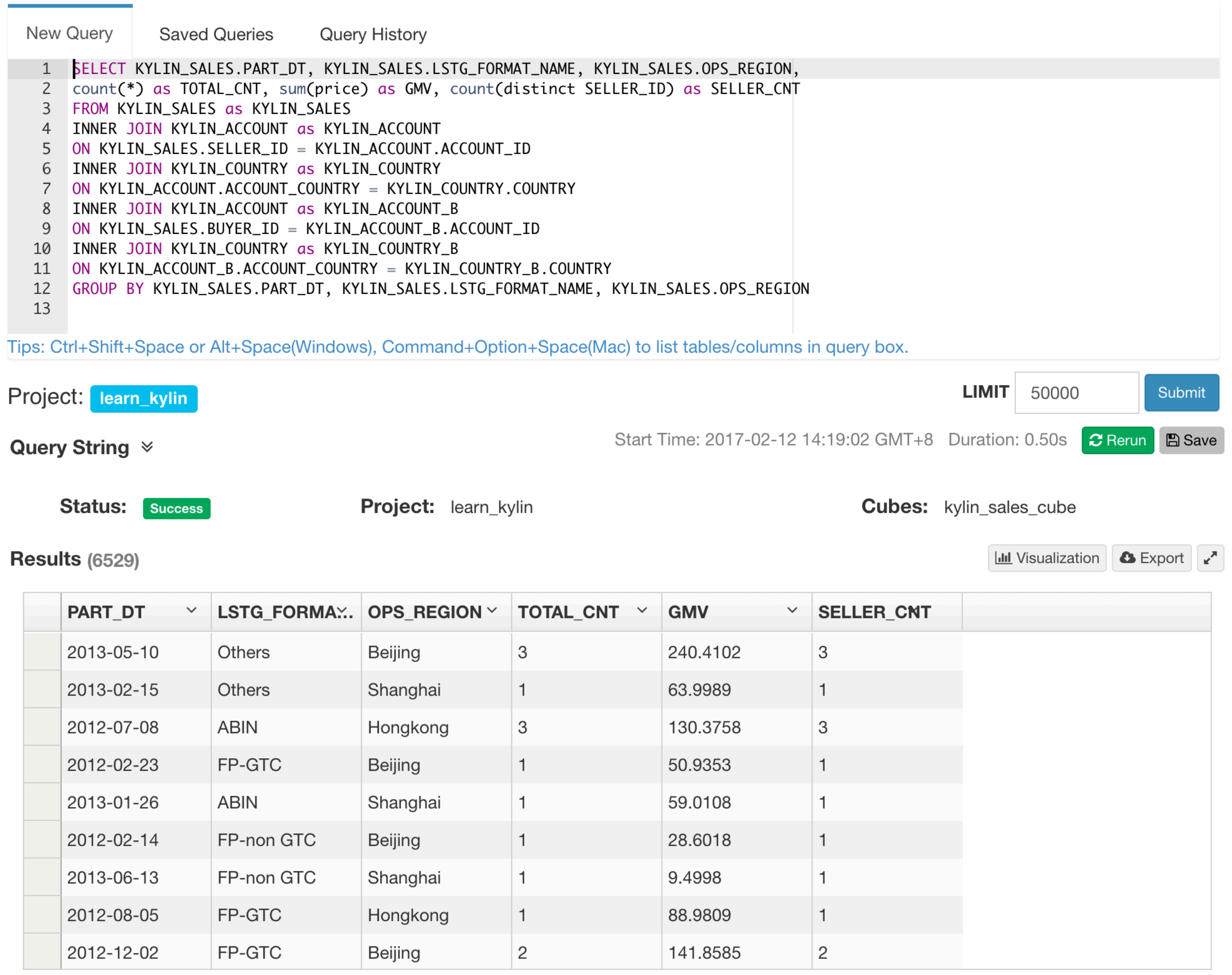
Segment: 20120101000000_20170213094500

- Start Time: 2012-01-01 00:00:00
- End Time: 2017-02-13 09:45:00
- Source Count: 10000
- HBase Table: KYLIN_UCXZNPB4PS
- Region Count: 3
- Size: 91 MB

Kyligence

查询

标准SQL接口



```sql
SELECT KYLIN_SALES.PART_DT, KYLIN_SALES.LSTG_FORMAT_NAME, KYLIN_SALES.OPS_REGION,
count(*) as TOTAL_CNT, sum(price) as GMV, count(distinct SELLER_ID) as SELLER_CNT
FROM KYLIN_SALES as KYLIN_SALES
INNER JOIN KYLIN_ACCOUNT as KYLIN_ACCOUNT
ON KYLIN_SALES.SELLER_ID = KYLIN_ACCOUNT.ACCOUNT_ID
INNER JOIN KYLIN_COUNTRY as KYLIN_COUNTRY
ON KYLIN_ACCOUNT.ACCOUNT_COUNTRY = KYLIN_COUNTRY.COUNTRY
INNER JOIN KYLIN_ACCOUNT as KYLIN_ACCOUNT_B
ON KYLIN_SALES.BUYER_ID = KYLIN_ACCOUNT_B.ACCOUNT_ID
INNER JOIN KYLIN_COUNTRY as KYLIN_COUNTRY_B
ON KYLIN_ACCOUNT_B.ACCOUNT_COUNTRY = KYLIN_COUNTRY_B.COUNTRY
GROUP BY KYLIN_SALES.PART_DT, KYLIN_SALES.LSTG_FORMAT_NAME, KYLIN_SALES.OPS_REGION
```

Tips: Ctrl+Shift+Space or Alt+Space(Windows), Command+Option+Space(Mac) to list tables/columns in query box.

Project: learn_kylin   LIMIT 50000   Submit

Query String   Start Time: 2017-02-12 14:19:02 GMT+8   Duration: 0.50s   Rerun   Save

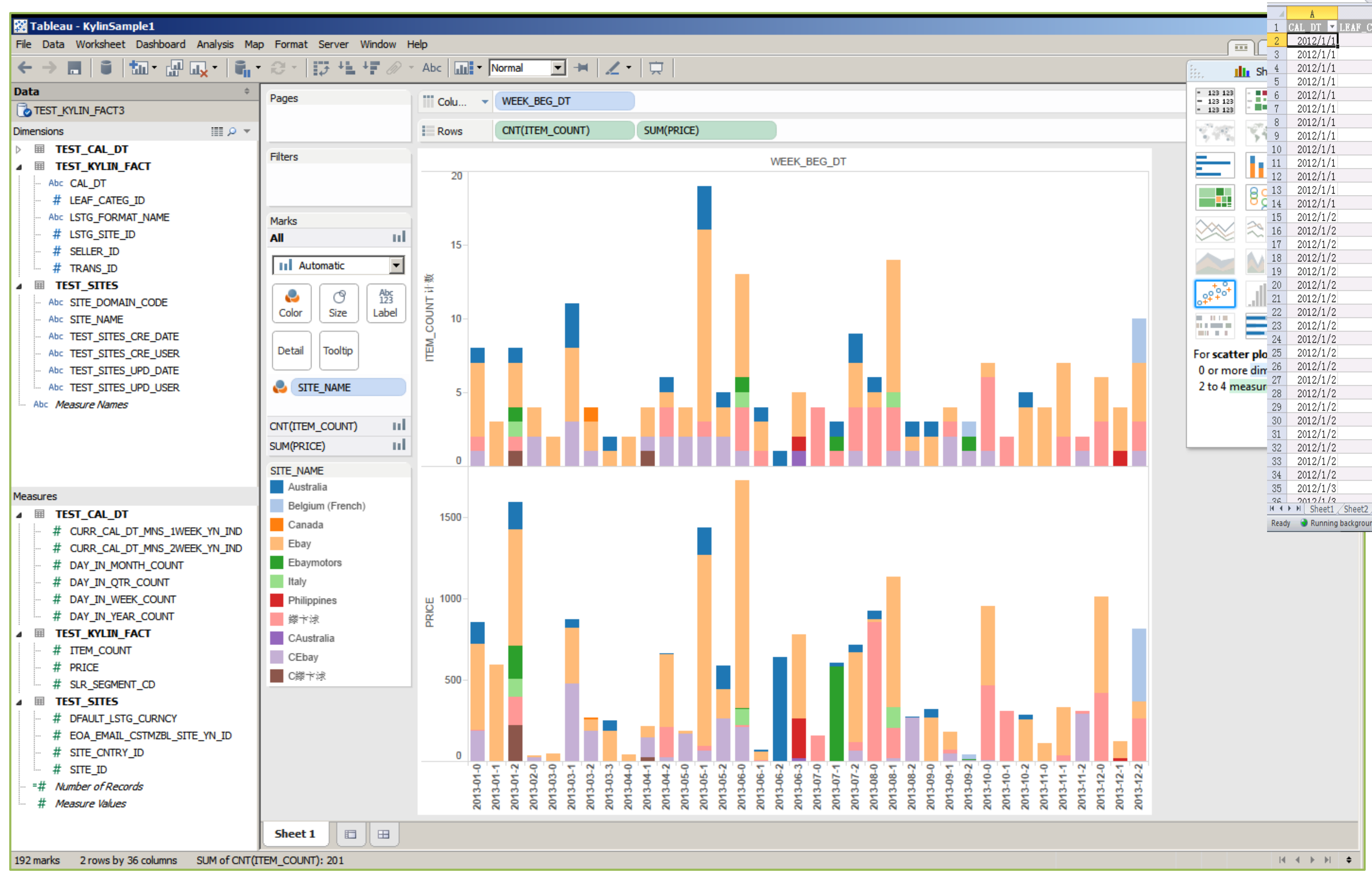
Status: Success   Project: learn_kylin   Cubes: kylin_sales_cube

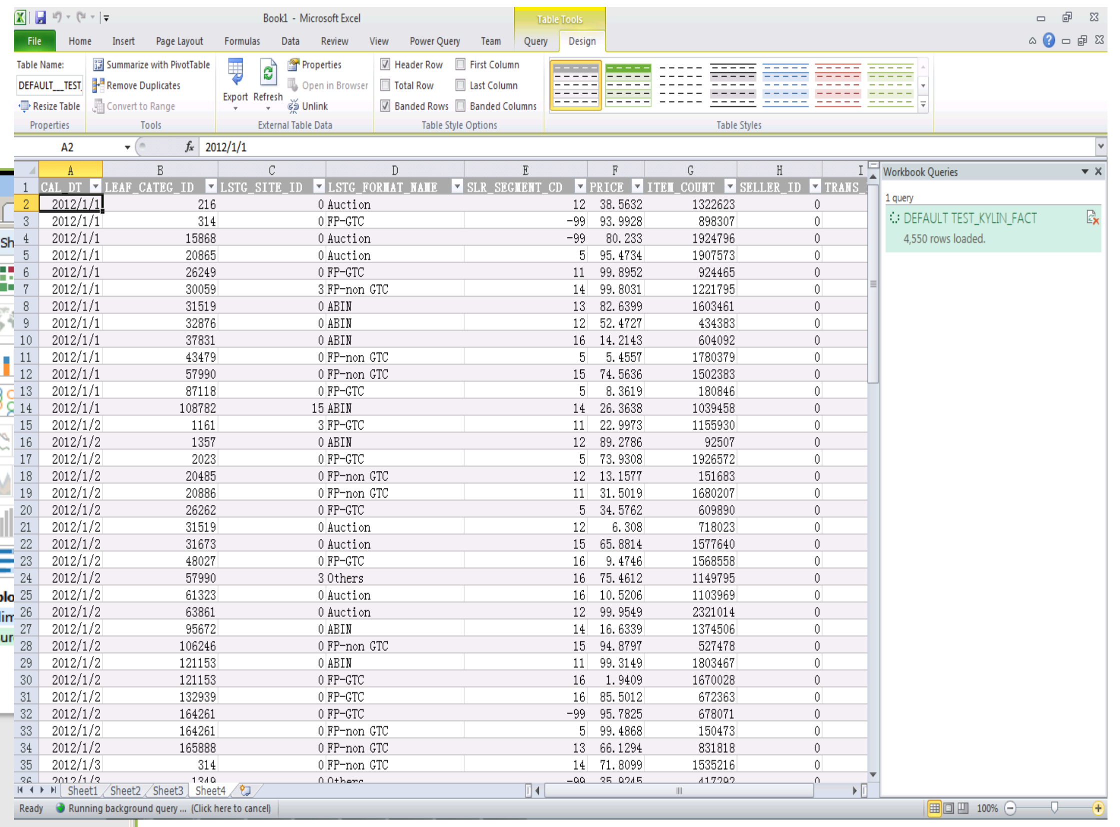
Results (6529)   Visualization   Export

PART_DT	LSTG_FORMA...	OPS_REGION	TOTAL_CNT	GMV	SELLER_CNT
2013-05-10	Others	Beijing	3	240.4102	3
2013-02-15	Others	Shanghai	1	63.9989	1
2012-07-08	ABIN	Hongkong	3	130.3758	3
2012-02-23	FP-GTC	Beijing	1	50.9353	1
2013-01-26	ABIN	Shanghai	1	59.0108	1
2012-02-14	FP-non GTC	Beijing	1	28.6018	1
2013-06-13	FP-non GTC	Shanghai	1	9.4998	1
2012-08-05	FP-GTC	Hongkong	1	88.9809	1
2012-12-02	FP-GTC	Beijing	2	141.8585	2

兼容BI工具（支持ODBC、JDBC、Rest API）

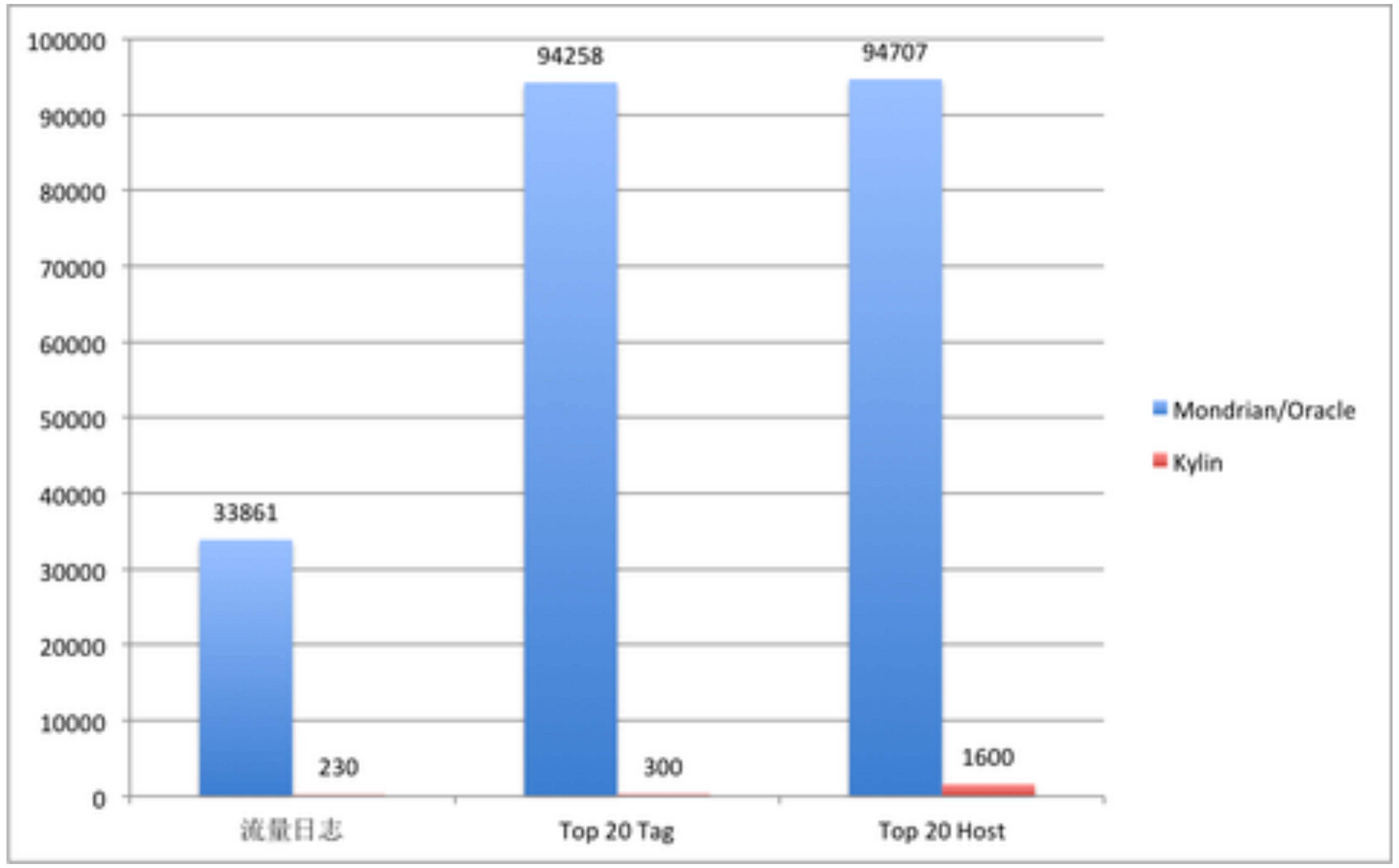
- Tableau
- Cognos BI
- Apache Zeppelin
- …
- Smart BI
- Excel / Power BI
- Caravel

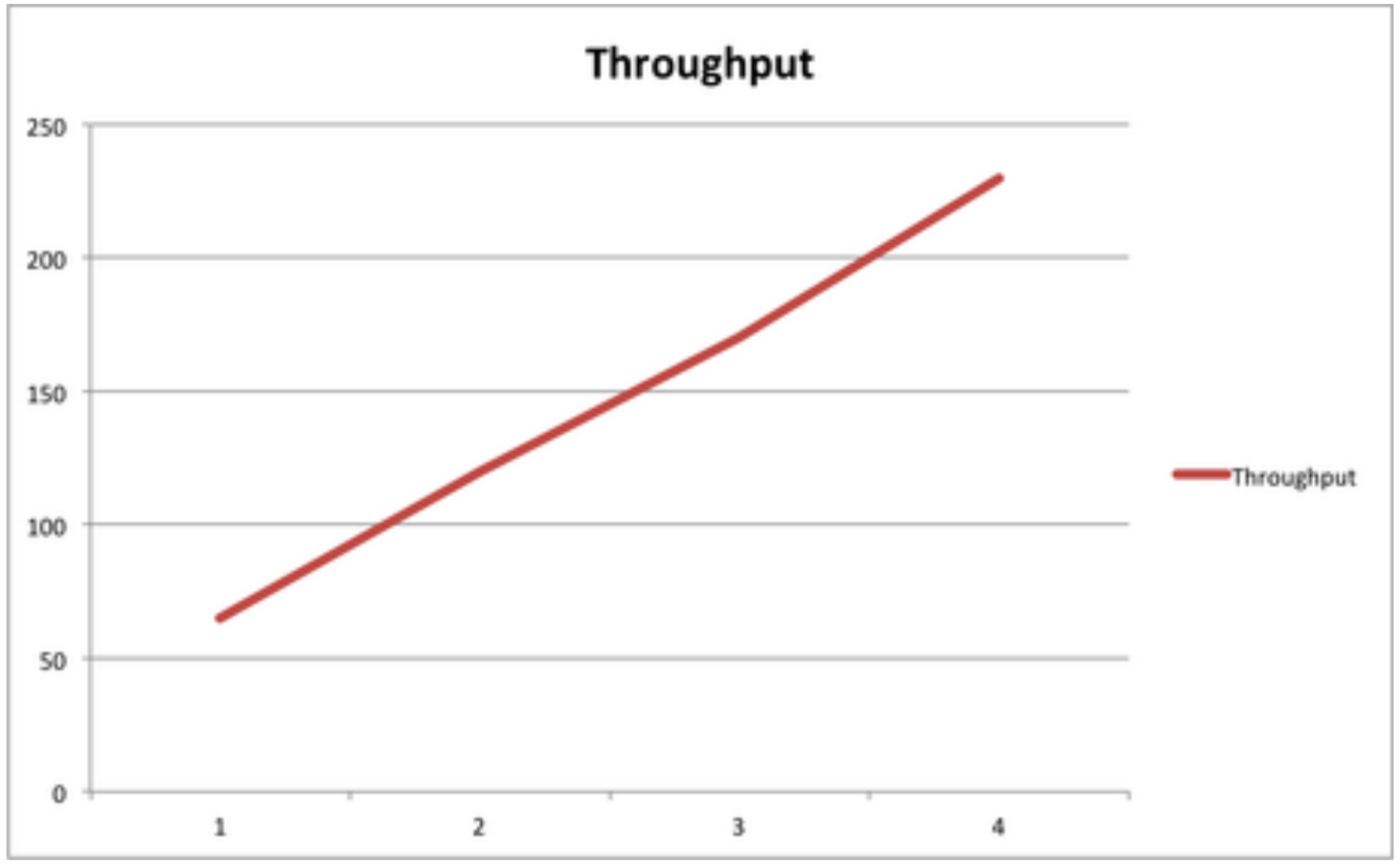




All rights reserved ©Kyligence Inc.
http://kyligence.io

性能和并发度





By 网易:
http://www.bitstech.net/2016/01/04/kylin-olap/

Apache Kylin全球用户

網易
美团 meituan.com
Baidu地图
唯品会 vip.com
JD.COM 京东
搜狐 SOHU.com
乐视视频 www.le.com
360 www.360.cn
去哪儿? Qunar.Com 聪明你的旅行
1号店
UnionPay 银联
中国移动 China Mobile
万里通 PINGAN
Lenovo
国美在线 GOME.COM.CN
国泰君安证券 GUOTAI JUNAN SECURITIES
科大讯飞 iFLYTEK
oppo
MEIZU
有赞 youzan.com
ebay
Bank of America
Microsoft
exponential Advertising Intelligence
Expedia
MZ MACHINE ZONE

- **eBay:**
  - Apache Kylin是目前最好的大数据OLAP引擎，当其他OLAP引擎与大数据量作斗争时，**Kylin仍能保持毫秒级响应**。此外，我们也开始使用Kylin作为**近实时数据流的存储和分析引擎**。总而言之，Kylin担任了eBay产品分析平台的关键后台组件。
- **京东**
  - Apache Kylin及其海量数据下使用标准SQL进行低延迟查询的特性，解决了在数据量极具增长的情况下**低延迟查询和平滑扩容**的挑战，通过对每天增量超过7亿条的API统计数据的分析，Apache Kylin使得我们在**百亿规模数据集上进行秒级多维分析**成为可能。

v2.0主要更新

OSC源创会 Opensource Innovation Meetup | IT大咖说

v1.5
- ••可插拔架构
- ••存储并行扫描
- ••Fast Cubing

v1.6
- ••支持流式数据

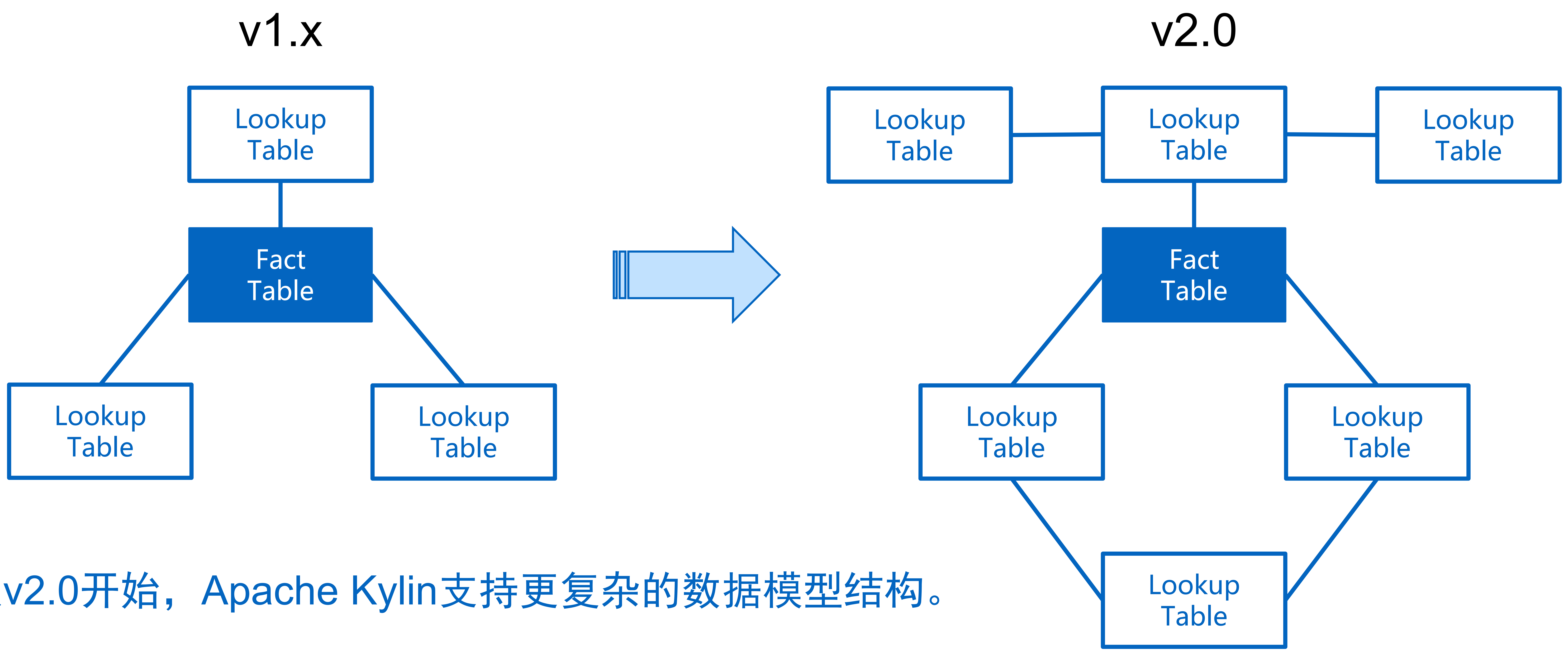
v2.0
- ••支持雪花模型
- ••Spark Cubing
- ••高级SQL函数
- ••…

Apache Kylin 2.0 (Beta)将于近期发布，请关注官网http://kylin.apache.org

Kyligence

支持雪花模型

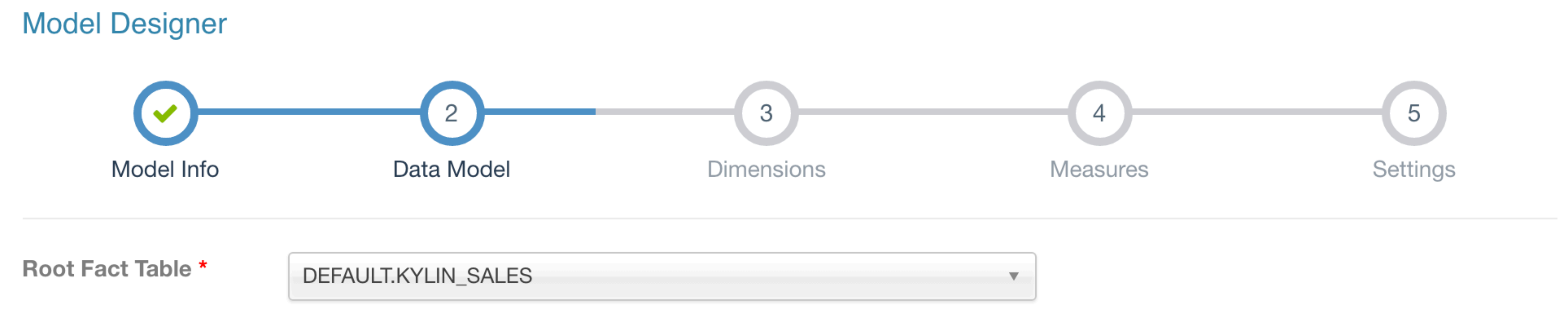
支持雪花模型



从v2.0开始，Apache Kylin支持更复杂的数据模型结构。

创建雪花模型

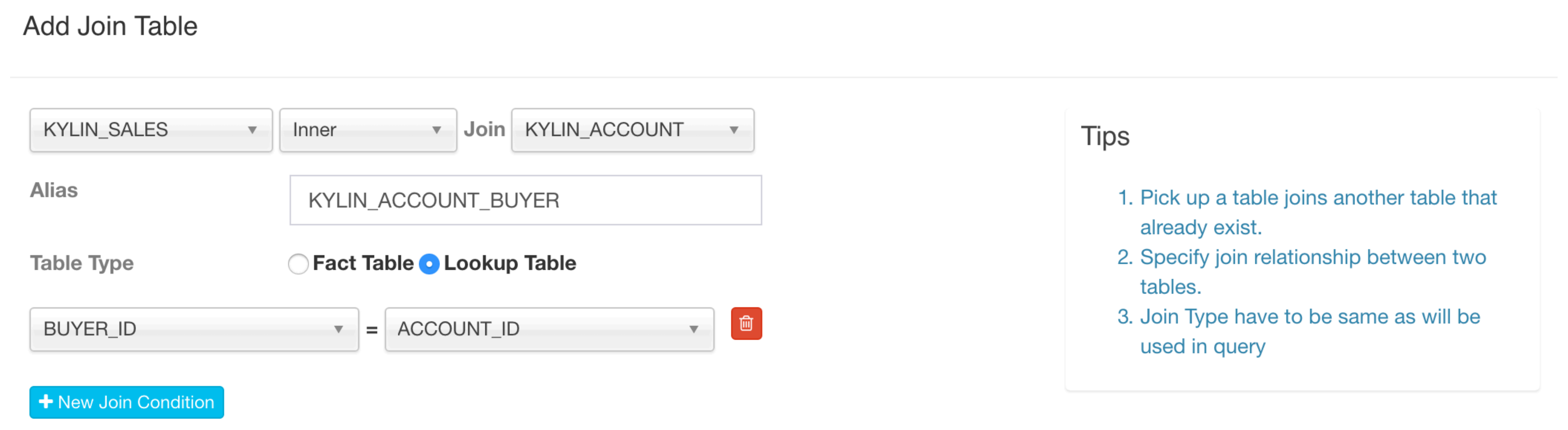
- **选择一个Fact表**


Model Designer

Model Info — Data Model — Dimensions — Measures — Settings

Root Fact Table *: DEFAULT.KYLIN_SALES

- **添加Lookup表，并设置Table Alias和Join条件**


Add Join Table

KYLIN_SALES | Inner | Join | KYLIN_ACCOUNT

Alias: KYLIN_ACCOUNT_BUYER

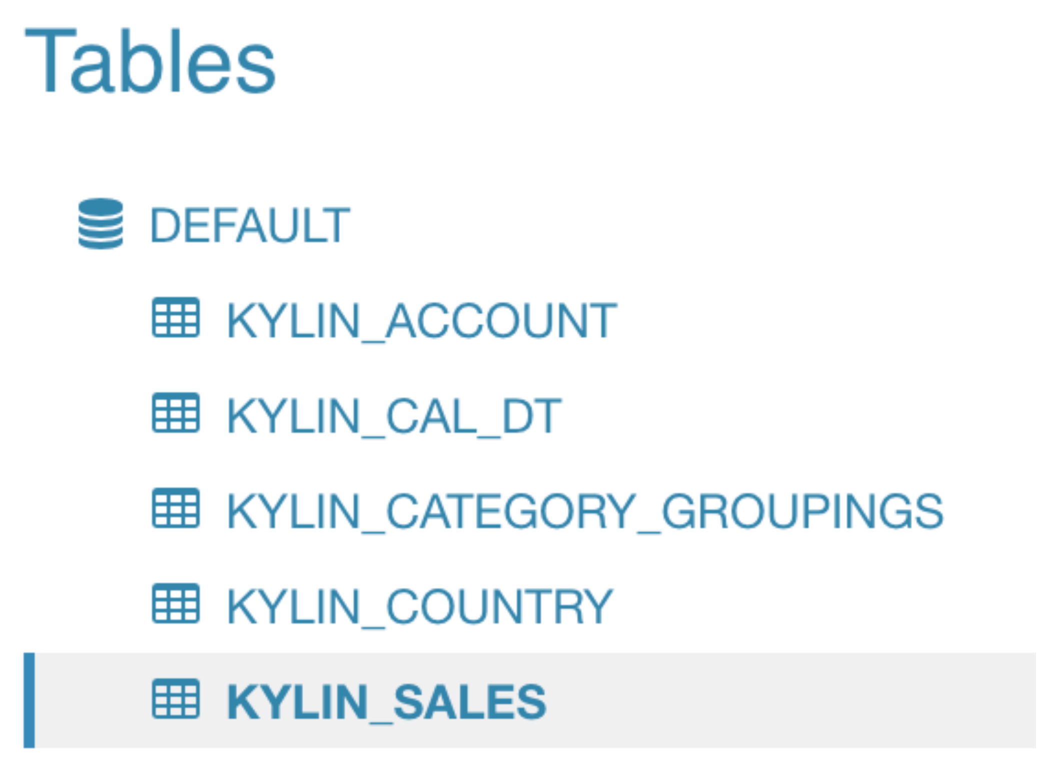
Table Type: Fact Table [✓ Lookup Table]

BUYER_ID = ACCOUNT_ID

New Join Condition

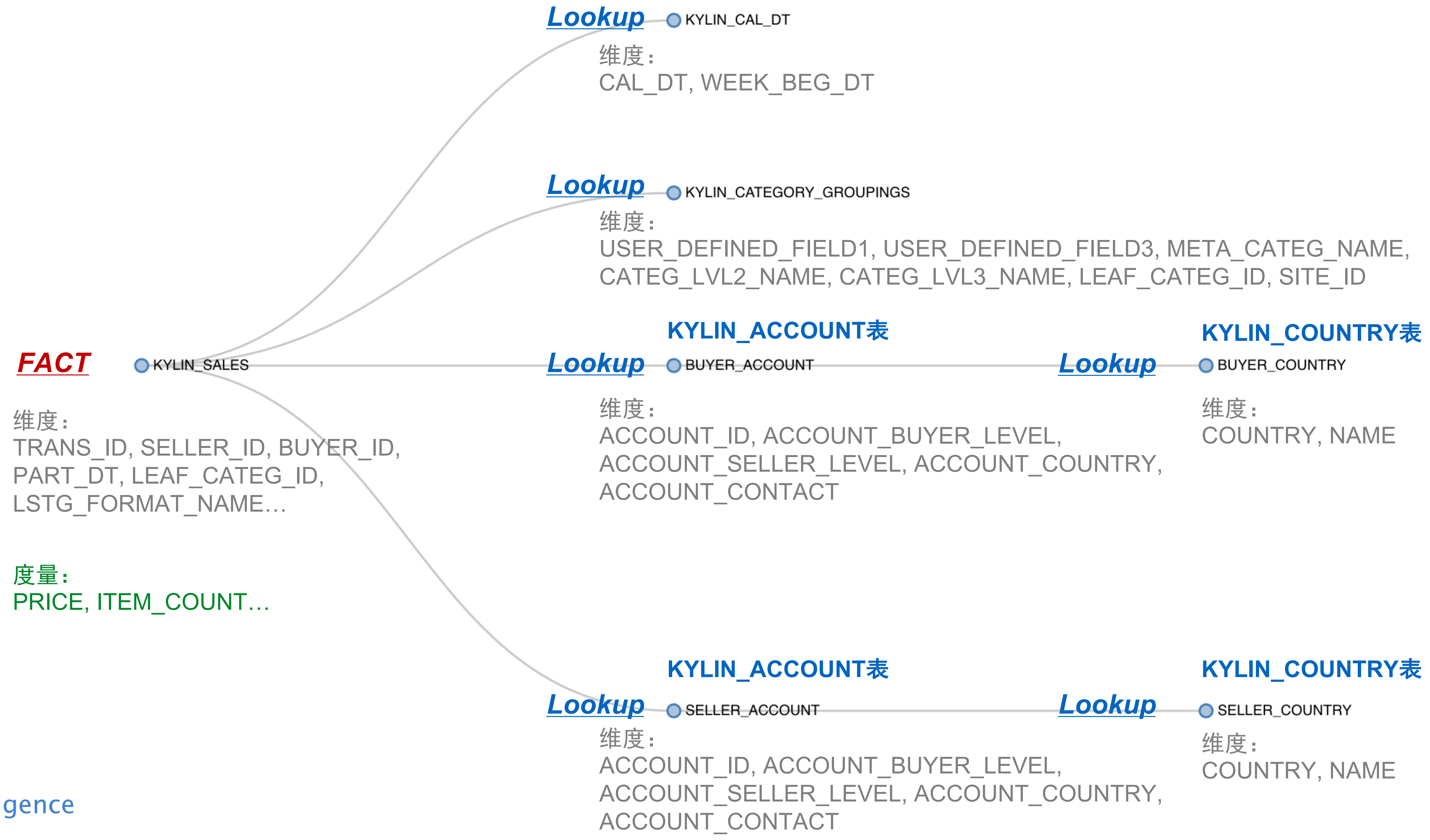
Tips

1. Pick up a table joins another table that already exist.
2. Specify join relationship between two tables.
3. Join Type have to be same as will be used in query


Tables

- DEFAULT
  - KYLIN_ACCOUNT
  - KYLIN_CAL_DT
  - KYLIN_CATEGORY_GROUPINGS
  - KYLIN_COUNTRY
  - **KYLIN_SALES**

雪花模型样例



FACT KYLIN_SALES

维度：
TRANS_ID, SELLER_ID, BUYER_ID, PART_DT, LEAF_CATEG_ID, LSTG_FORMAT_NAME…

度量：
PRICE, ITEM_COUNT…

Lookup KYLIN_CAL_DT

维度：
CAL_DT, WEEK_BEG_DT

Lookup KYLIN_CATEGORY_GROUPINGS

维度：
USER_DEFINED_FIELD1, USER_DEFINED_FIELD3, META_CATEG_NAME, CATEG_LVL2_NAME, CATEG_LVL3_NAME, LEAF_CATEG_ID, SITE_ID

KYLIN_ACCOUNT表

Lookup BUYER_ACCOUNT

维度：
ACCOUNT_ID, ACCOUNT_BUYER_LEVEL, ACCOUNT_SELLER_LEVEL, ACCOUNT_COUNTRY, ACCOUNT_CONTACT

KYLIN_COUNTRY表

Lookup BUYER_COUNTRY

维度：
COUNTRY, NAME

KYLIN_ACCOUNT表

Lookup SELLER_ACCOUNT

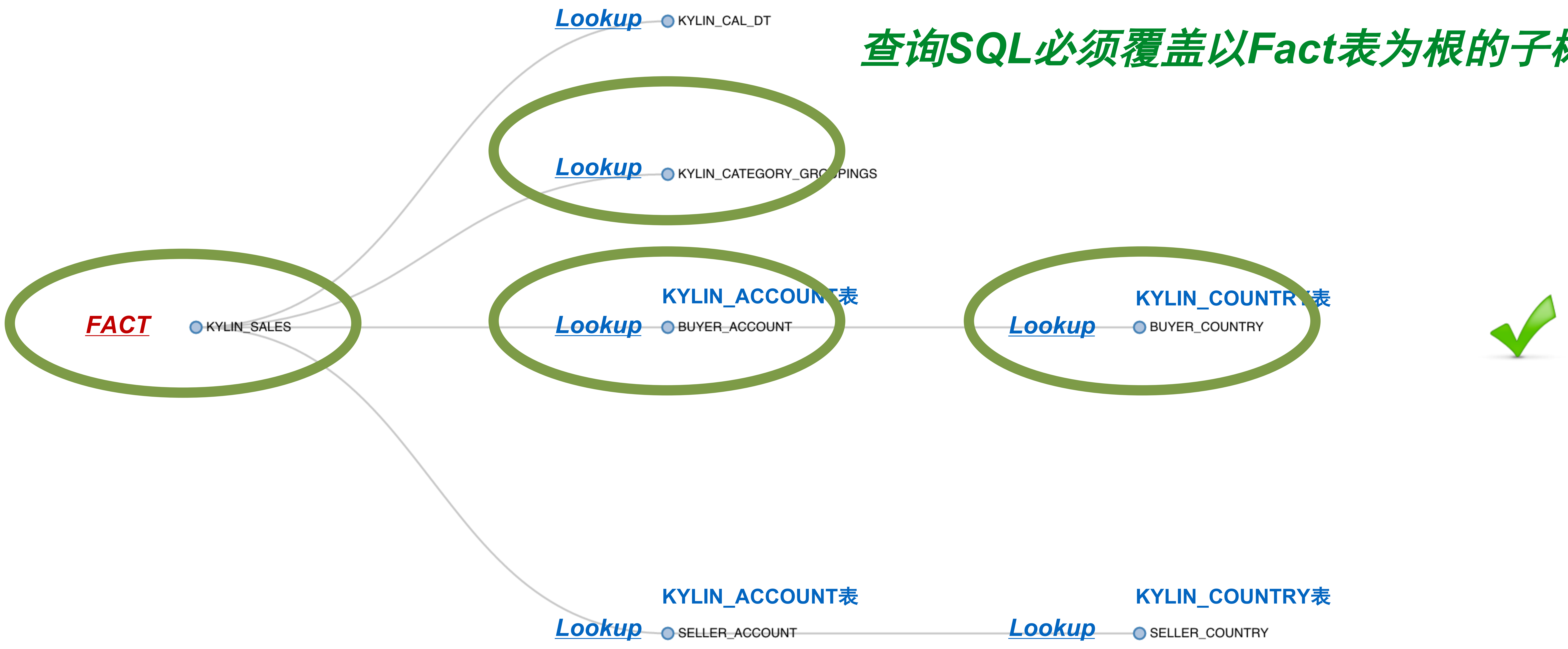
维度：
ACCOUNT_ID, ACCOUNT_BUYER_LEVEL, ACCOUNT_SELLER_LEVEL, ACCOUNT_COUNTRY, ACCOUNT_CONTACT

KYLIN_COUNTRY表

Lookup SELLER_COUNTRY

维度：
COUNTRY, NAME

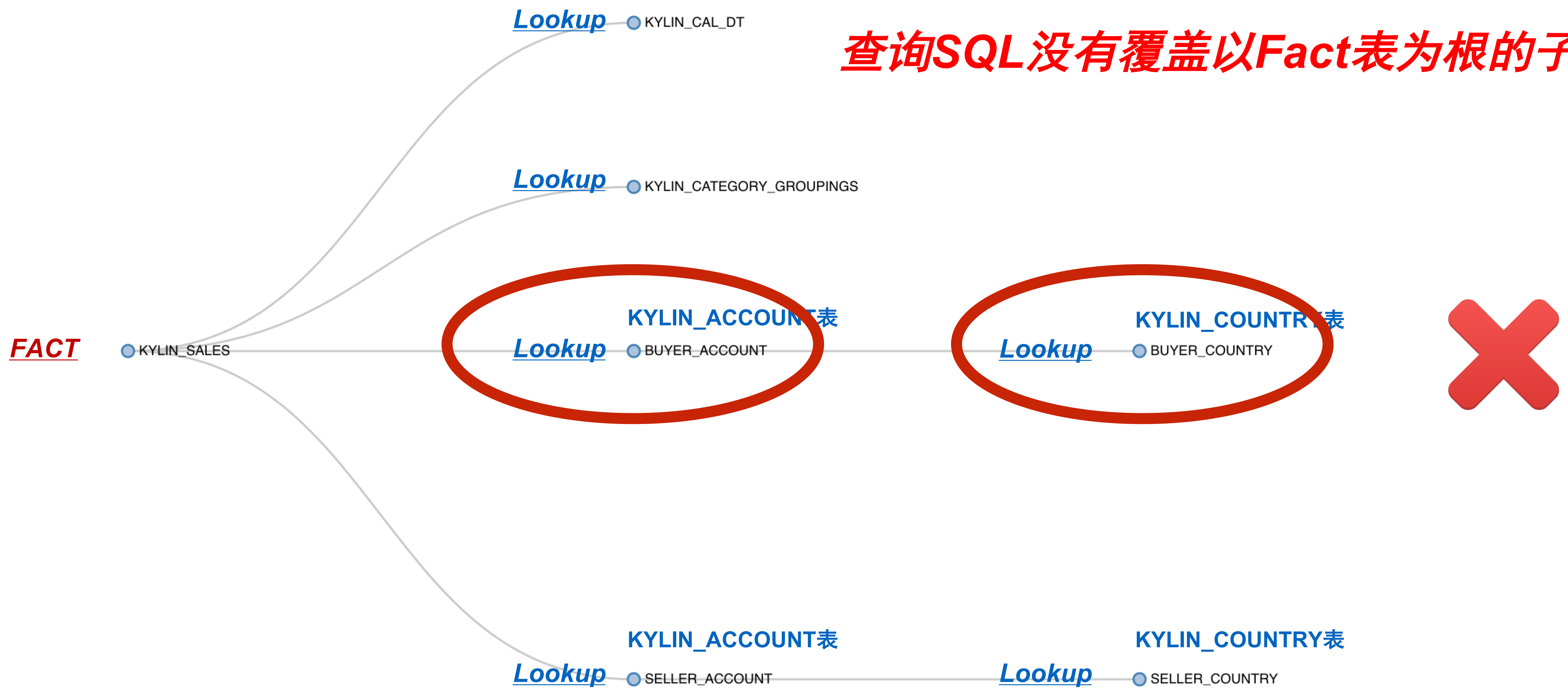
查询匹配



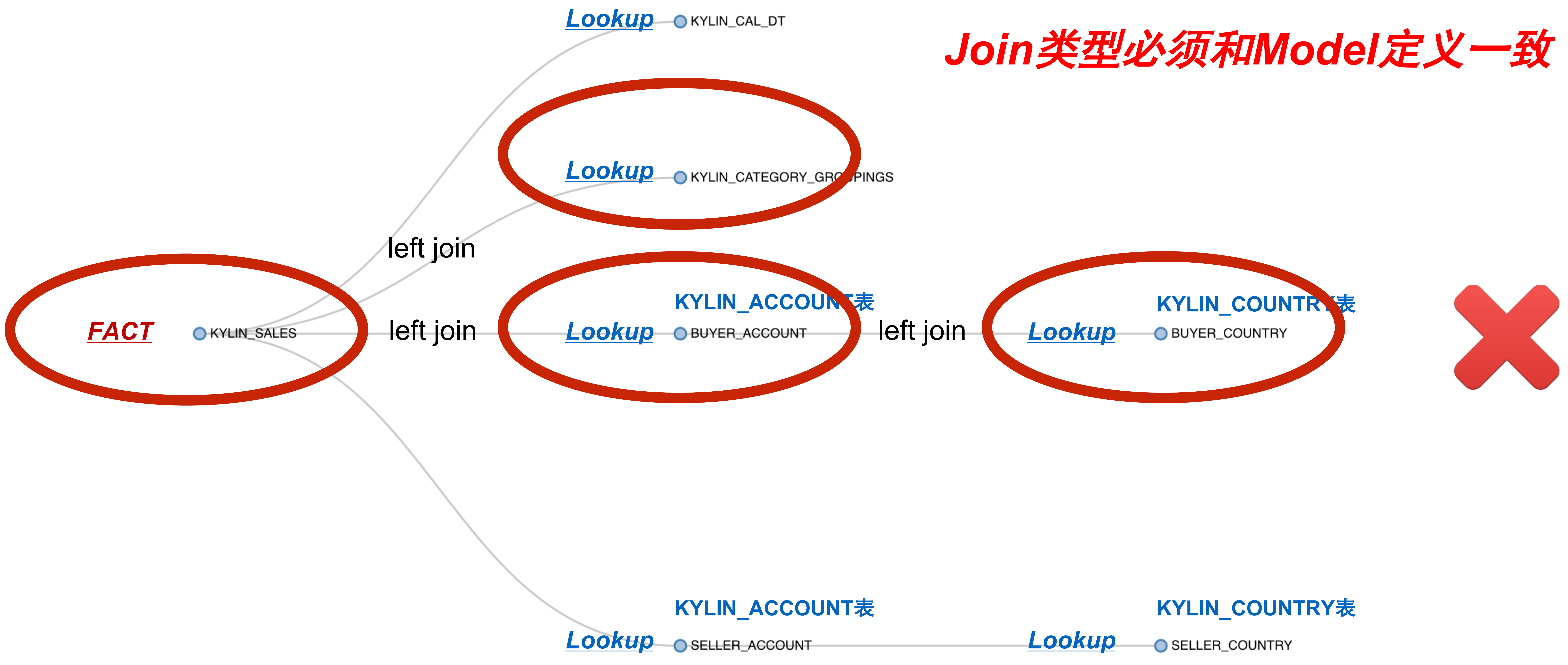
查询SQL必须覆盖以Fact表为根的子树

查询匹配

查询SQL没有覆盖以Fact表为根的子树



查询匹配


Lookup
KYLIN_CAL_DT
Join类型必须和Model定义一致
Lookup
KYLIN_CATEGORY_GROUPINGS
left join
FACT
KYLIN_SALES
left join
KYLIN_ACCOUNT表
Lookup
BUYER_ACCOUNT
left join
KYLIN_COUNTRY表
Lookup
BUYER_COUNTRY
KYLIN_ACCOUNT表
Lookup
SELLER_ACCOUNT
KYLIN_COUNTRY表
Lookup
SELLER_COUNTRY

查询局限

- **No raw records limitation.**

  select * from F $\longrightarrow$ select * from F group by D1, D2, ..., Dn

  - To support query of raw records, user can add PK as dimension (at the cost that the cardinality of PK is very high). Or try the raw measure feature.

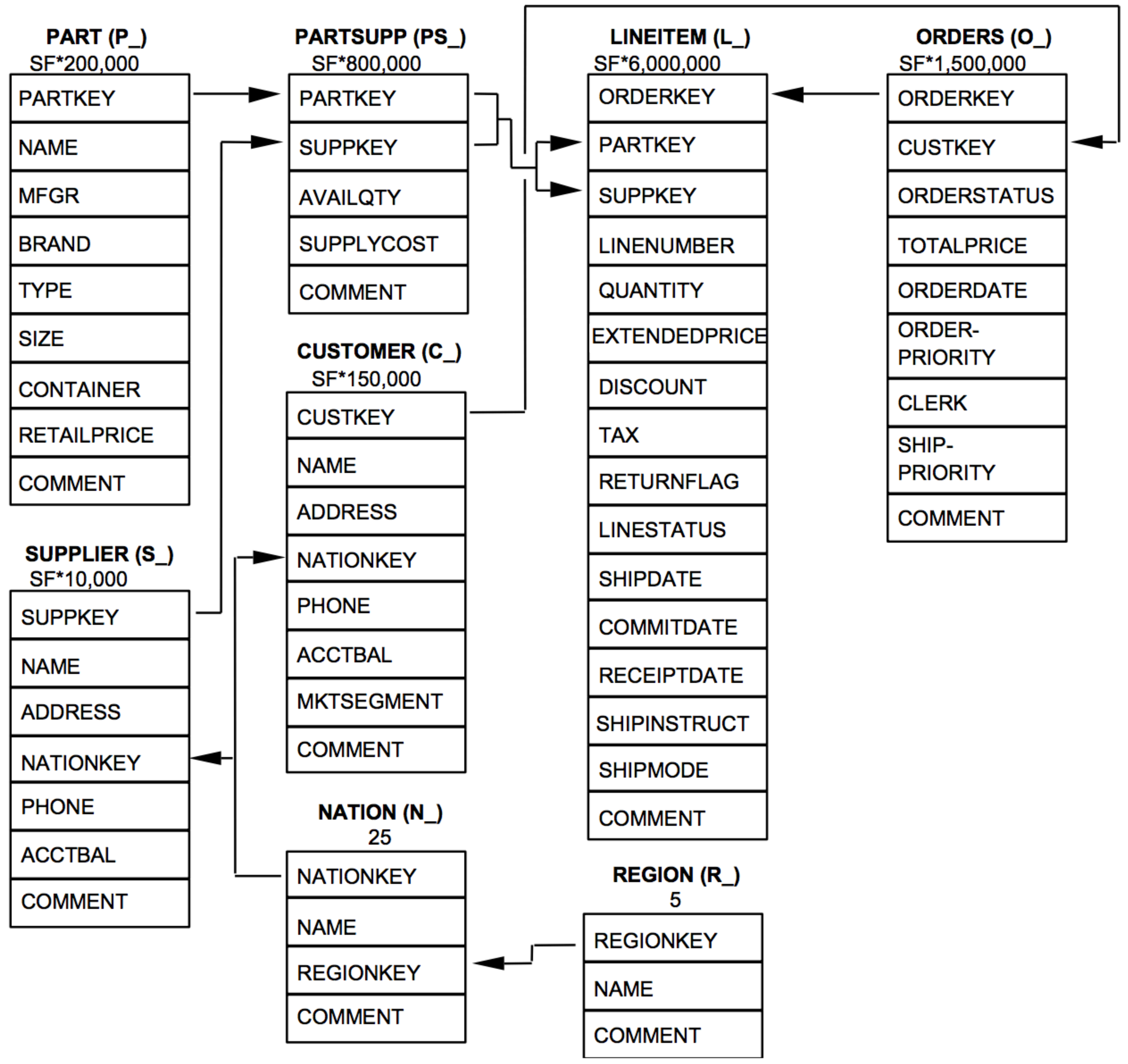
- **Enforced joint limitation.**

  select * from F $\longrightarrow$ select ... from F1 join L1 join L2...

  - This limitation IS NOT a problem for left join models and inner join models that has no record loss after inner join.
  - This limitation is a problem for many-to-many relationships. User can work around by creating multiple models, for example let each fact table has its own model.

TPC-H 基准测试

- 根据真实环境建模
- 数据仓库测试
- 22个查询

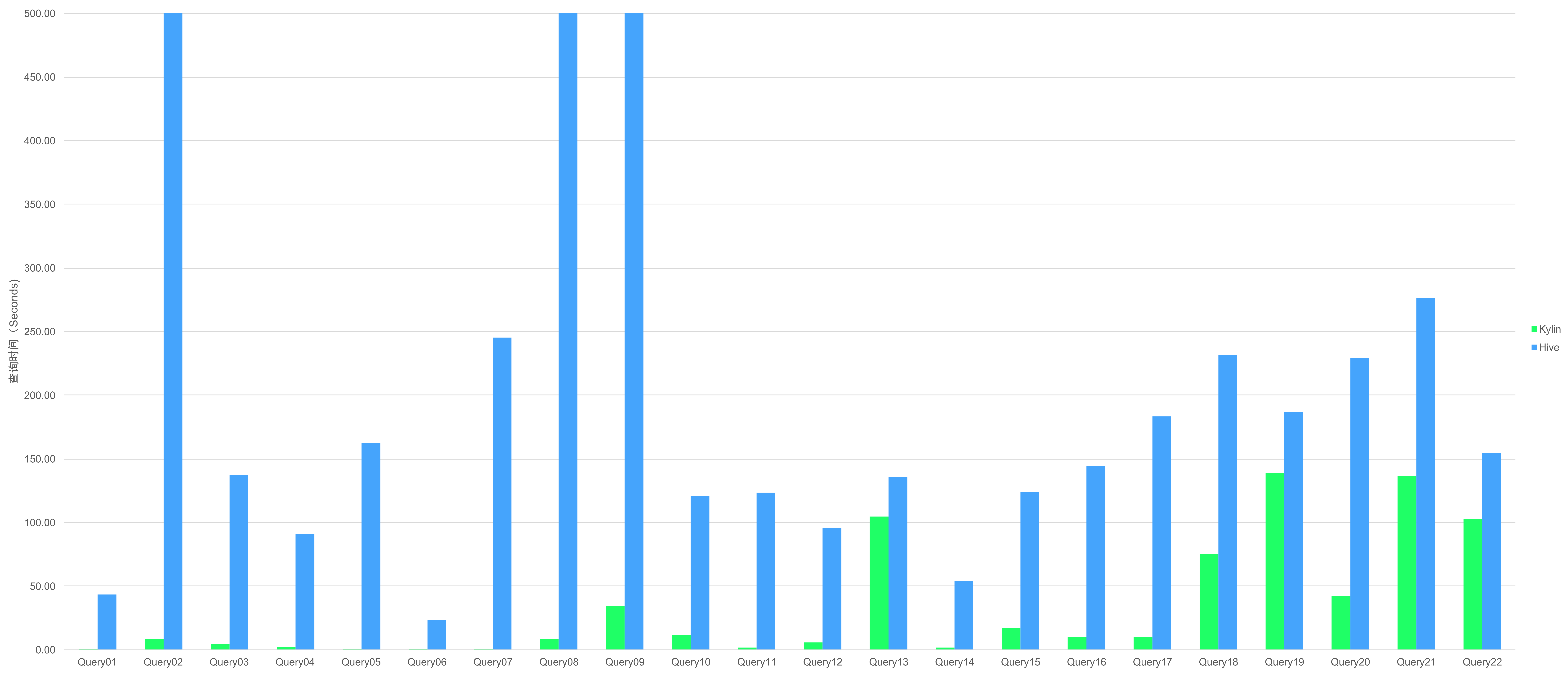


Cube 构建

- Cloudera 5.8.3
- 1 Master (32 vcores, 192 GB)
- 3 Slave (32 vcores, 96 GB)
- Scale factor=10

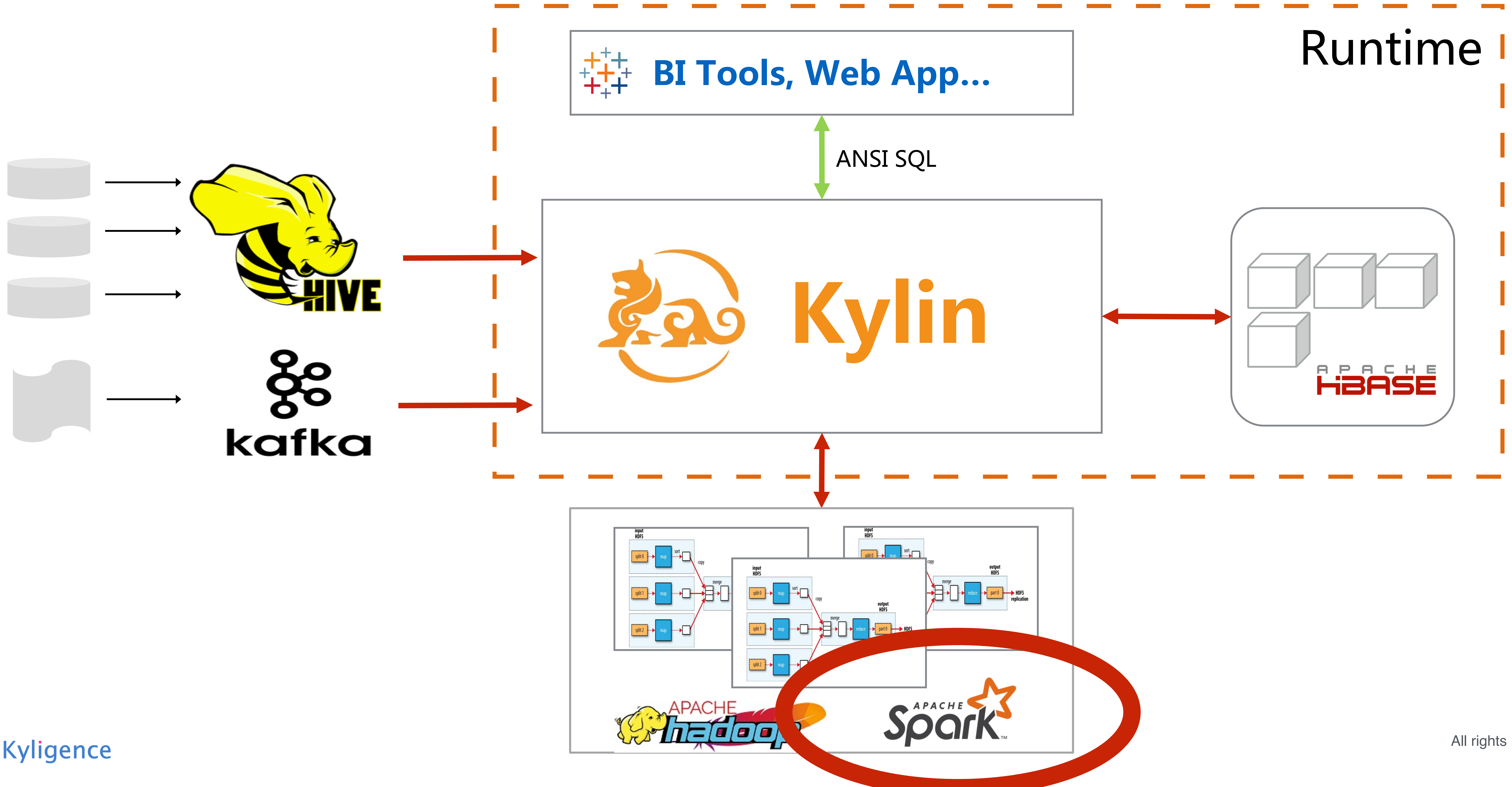
Cube	Build Stats
lineitem_cube	85.8 GB, 54.69 mins * 7
partsupp_cube	8.3 GB, 18.48 mins
customer_vorder_cube	1.39 GB, 13.7 mins
customer_cube	36.98 MB, 3.97 mins

查询：Kylin vs Hive



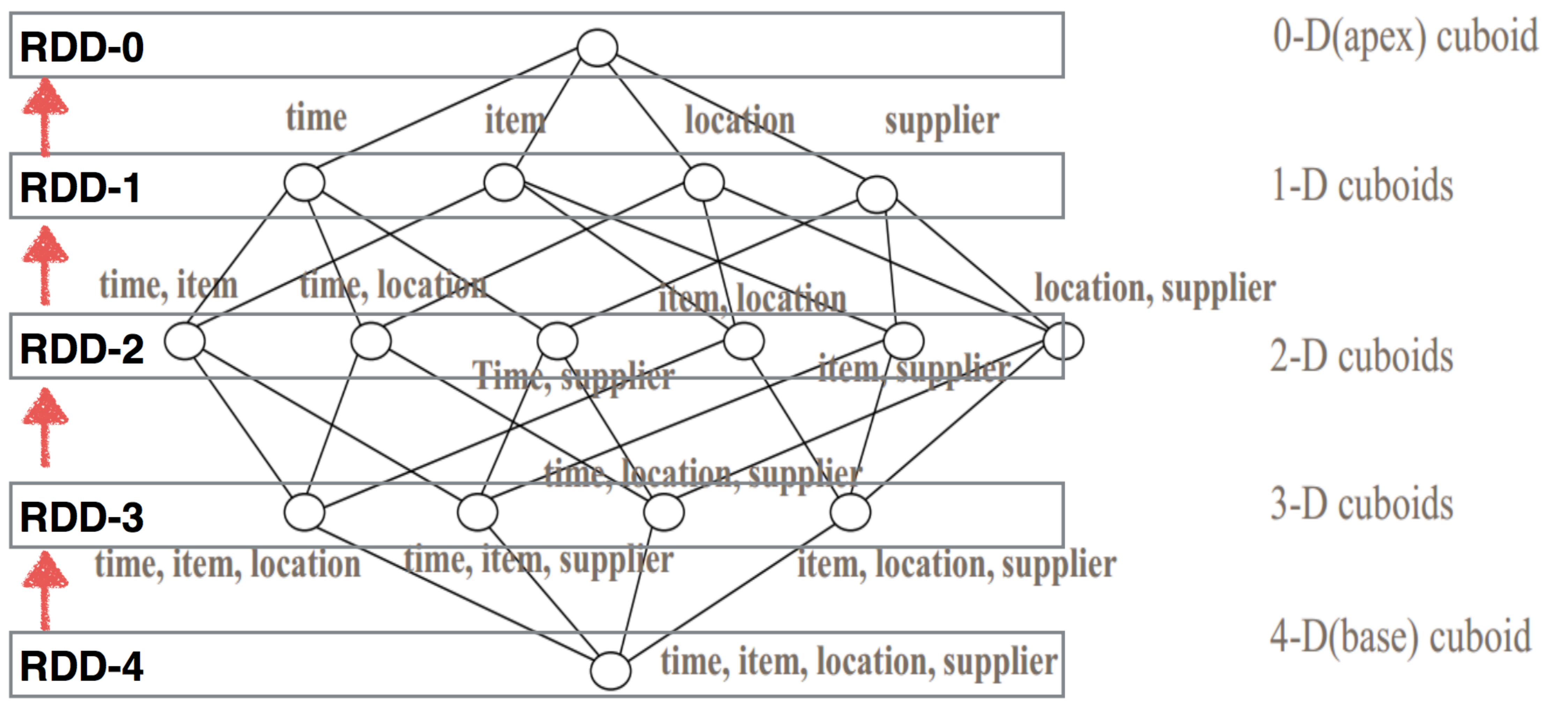
Spark 构建引擎

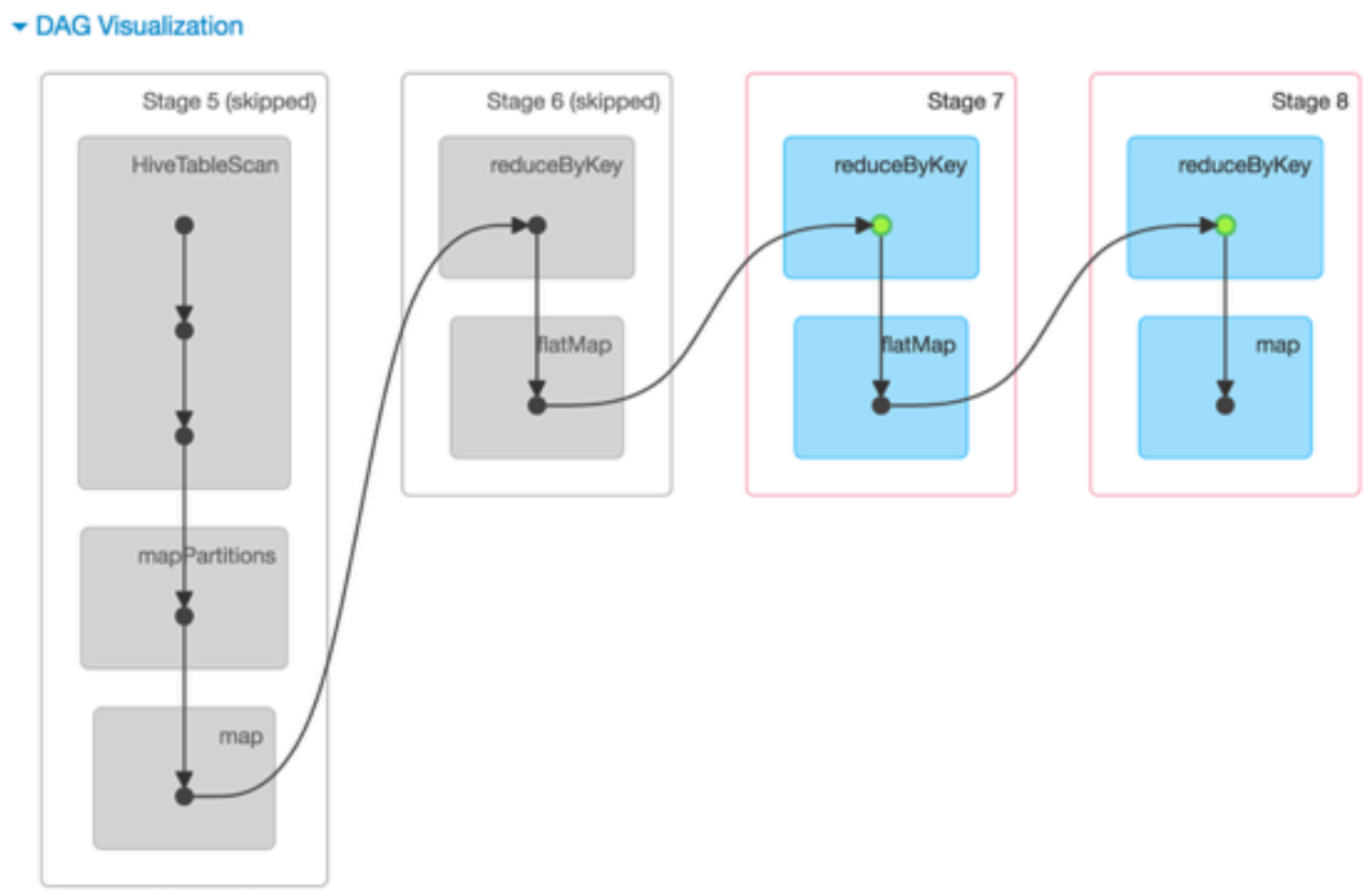
技术架构



Kyligence

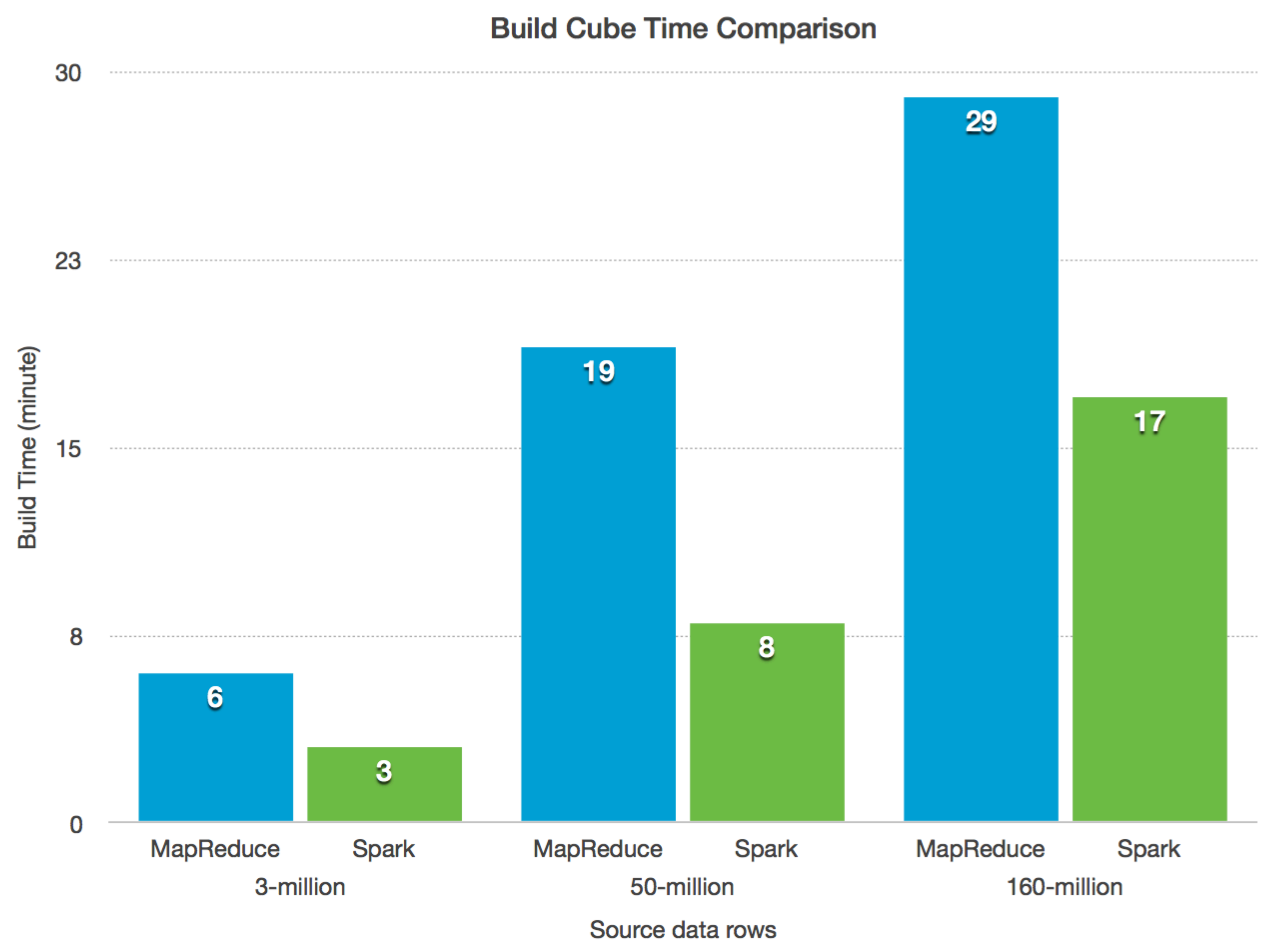
Spark Cubing





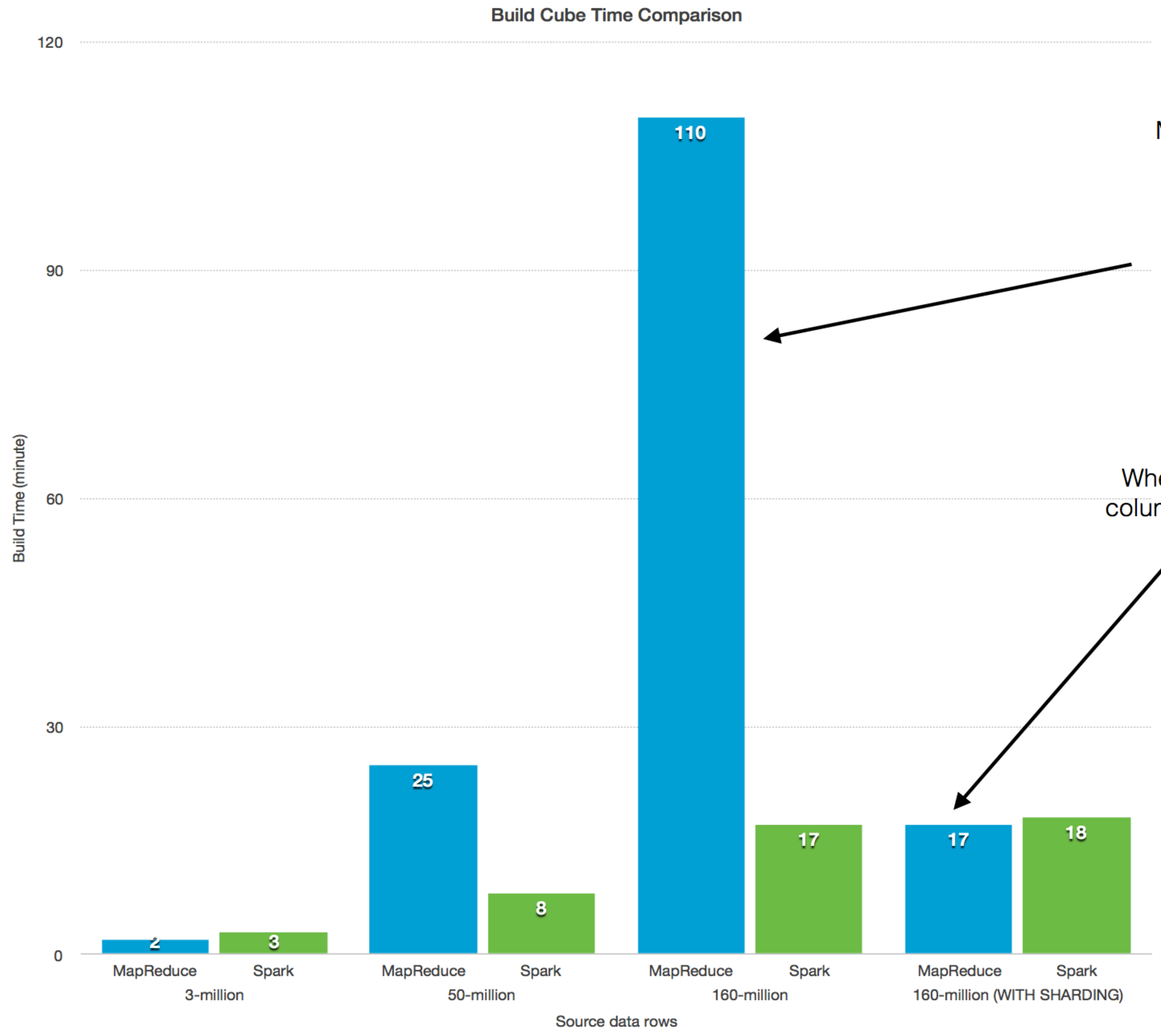
- Abstract each layer cuboids as an RDD
- A N-dim Cube will has N+1 RDDs
- Parent RDD can be cached when generating children RDDs
- RDD can be output to Sequence files in the same format as MR
- Translate MR's "map" and "reduce" to Spark's "flatMap" and "reduceByKey"; most codes get reused.

MR (by-layer) vs Spark



- By-layer cubing algorithm has stable performance on both MR and Spark.
- Compared with MR layered algorithm, Spark cubing has **70% to 130%** performance improvement.

MR (Fast Cubing) vs Spark



MR(In-Mem) is unstable when source data is random distributed (cause too many shuffles); When data is distributed by a column, it is pretty fast

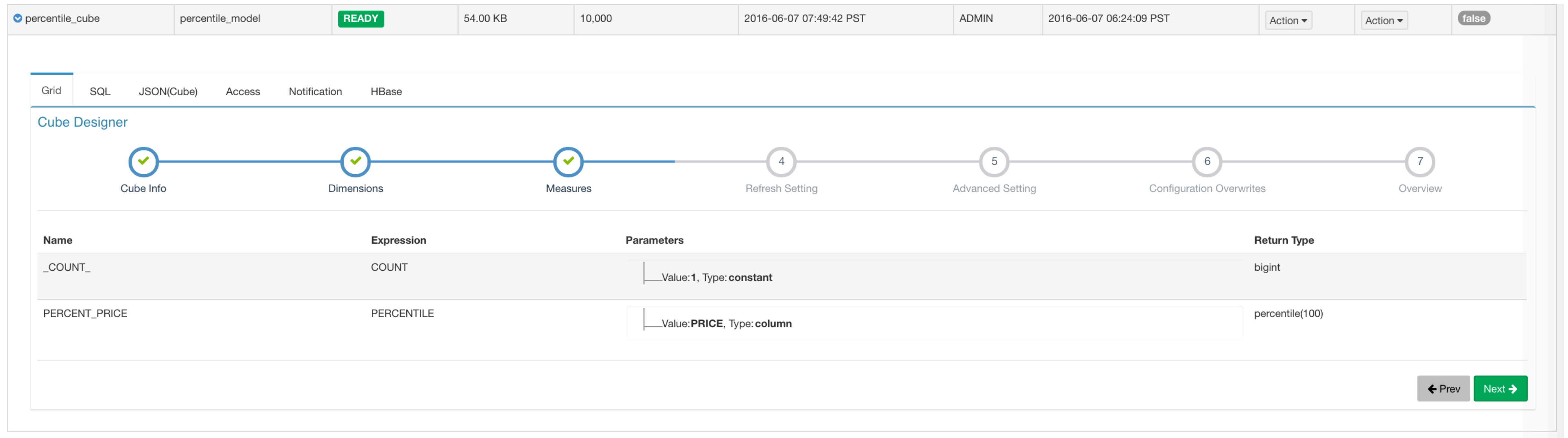
When data is distributed by a column, MR(In-Mem) is very fast

高级SQL支持

percentile

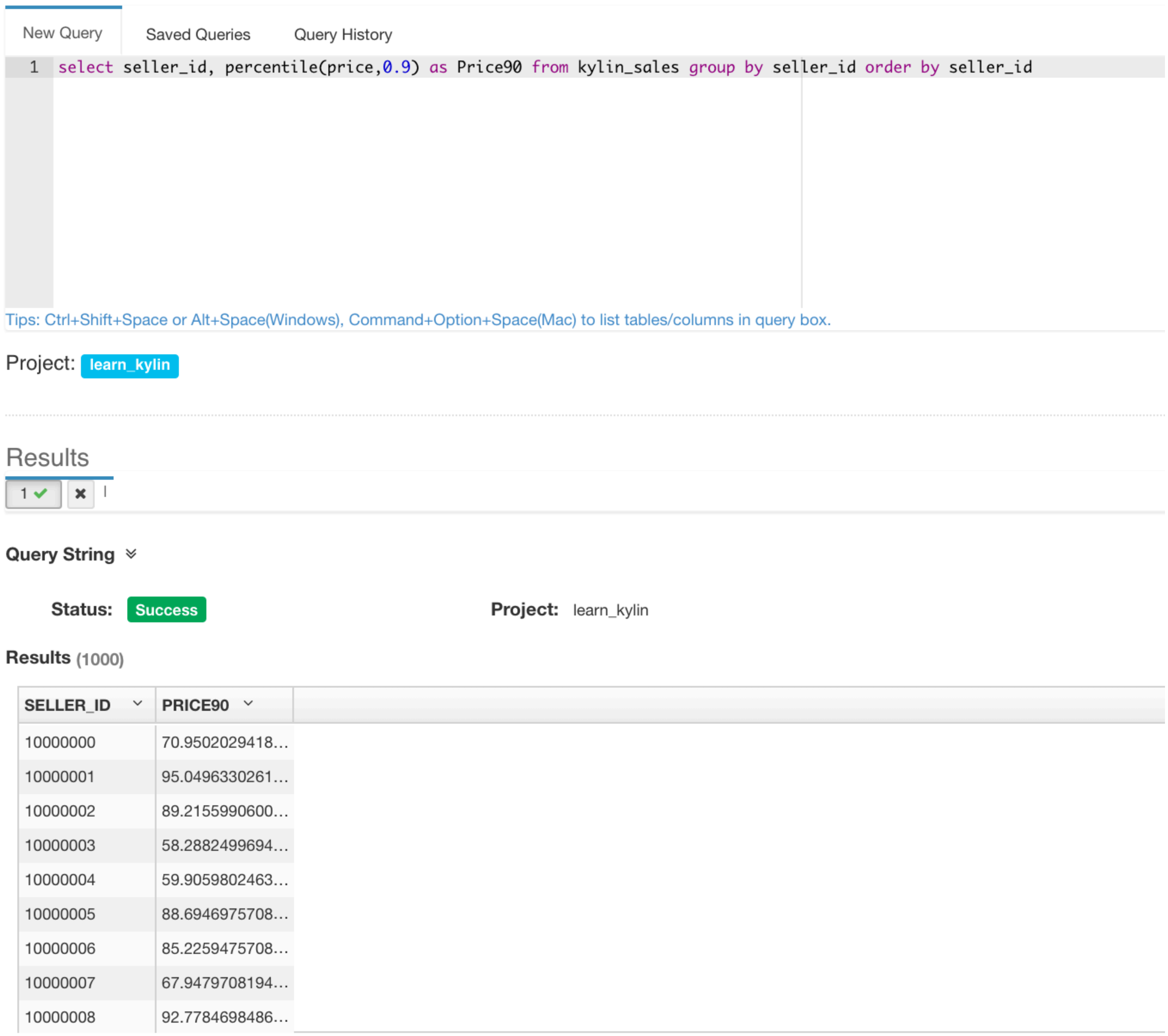
增加了一种计算百分位的度量：

percentile(price, 0.9)



percentile_cube	percentile_model	READY	54.00 KB	10,000	2016-06-07 07:49:42 PST	ADMIN	2016-06-07 06:24:09 PST	Action	Action	false

Name	Expression	Parameters	Return Type
COUNT	COUNT	Value:**1**, Type:**constant**	bigint
PERCENT_PRICE	PERCENTILE	Value:**PRICE**, Type:**column**	percentile(100)



```
select seller_id, percentile(price,0.9) as Price90 from kylin_sales group by seller_id order by seller_id
```

Tips: Ctrl+Shift+Space or Alt+Space(Windows), Command+Option+Space(Mac) to list tables/columns in query box.

Project: learn_kylin

Results

Query String

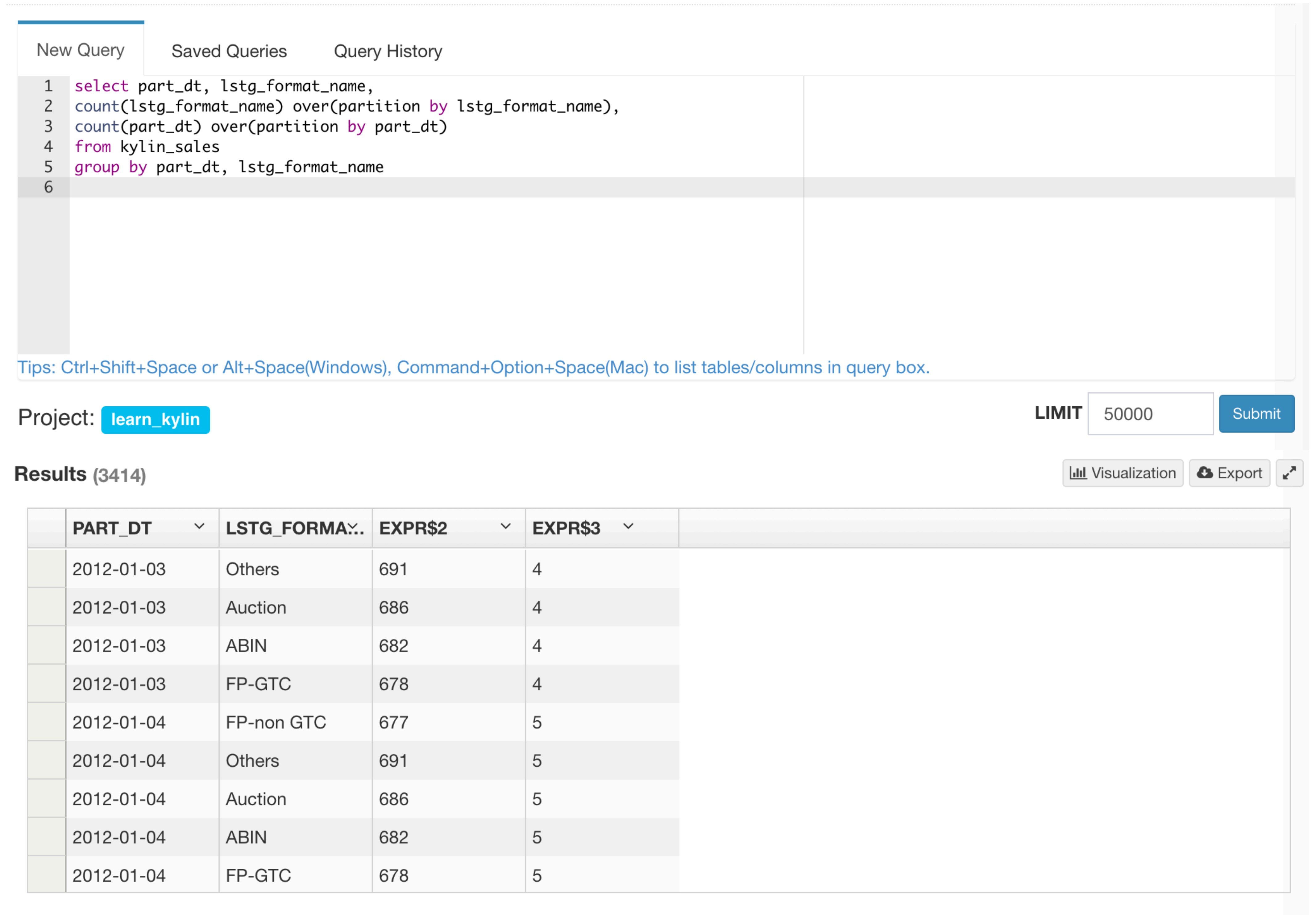
Status: Success   Project: learn_kylin

Results (1000)

SELLER_ID	PRICE90
10000000	70.9502029418…
10000001	95.0496330261…
10000002	89.2155990600…
10000003	58.2882499694…
10000004	59.9059802463…
10000005	88.6946975708…
10000006	85.2259475708…
10000007	67.9479708194…
10000008	92.7784698486…

窗口函数

Kylin v2.0支持多种窗口函数，以应对复杂的分析场景



```
select part_dt, lstg_format_name,
count(lstg_format_name) over(partition by lstg_format_name),
count(part_dt) over(partition by part_dt)
from kylin_sales
group by part_dt, lstg_format_name
```

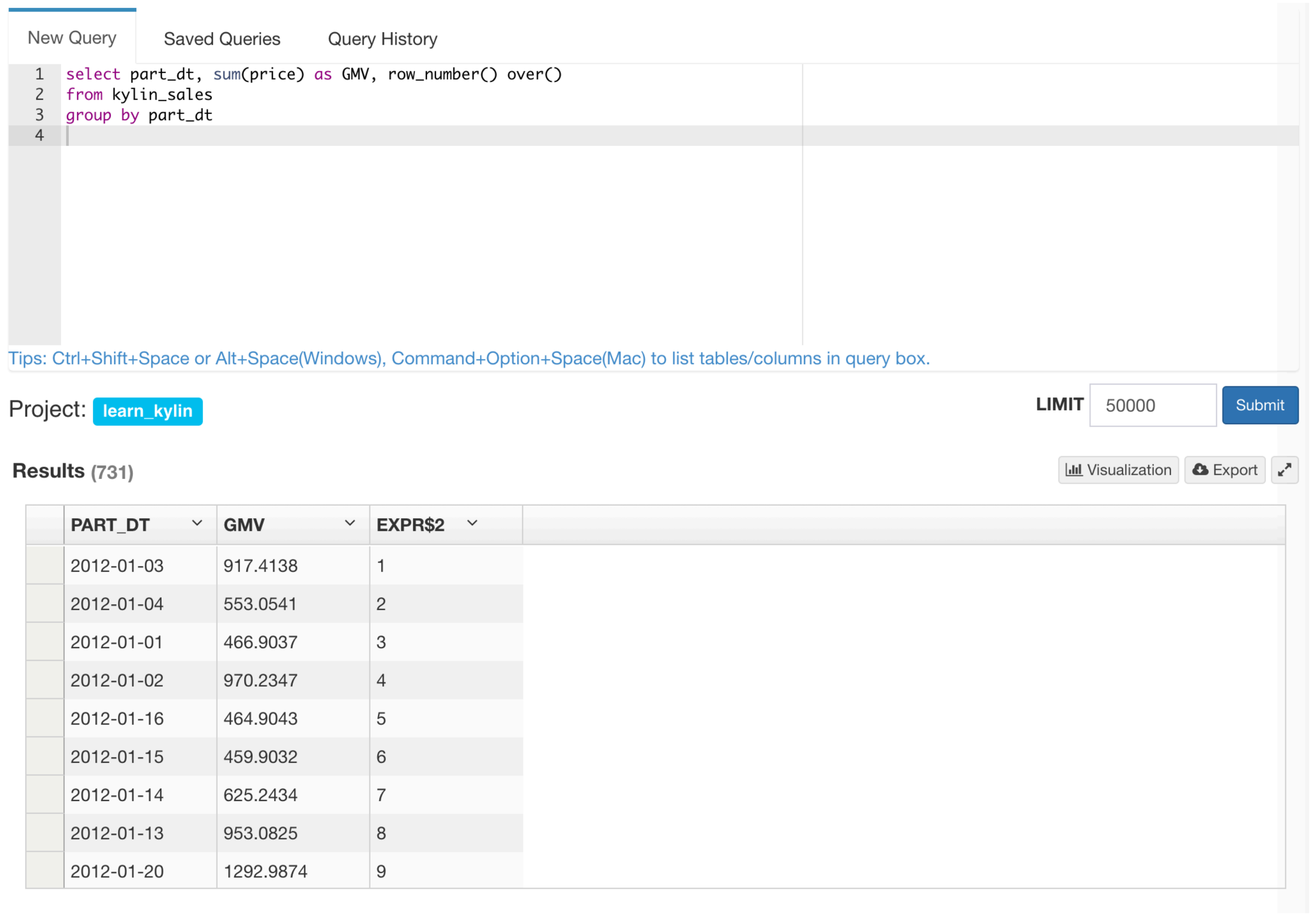
Tips: Ctrl+Shift+Space or Alt+Space(Windows), Command+Option+Space(Mac) to list tables/columns in query box.

Project: learn_kylin

LIMIT 50000

Results (3414)

PART_DT	LSTG_FORMA...	EXPR$2	EXPR$3
2012-01-03	Others	691	4
2012-01-03	Auction	686	4
2012-01-03	ABIN	682	4
2012-01-03	FP-GTC	678	4
2012-01-04	FP-non GTC	677	5
2012-01-04	Others	691	5
2012-01-04	Auction	686	5
2012-01-04	ABIN	682	5
2012-01-04	FP-GTC	678	5



```
select part_dt, sum(price) as GMV, row_number() over()
from kylin_sales
group by part_dt
```

Tips: Ctrl+Shift+Space or Alt+Space(Windows), Command+Option+Space(Mac) to list tables/columns in query box.

Project: learn_kylin

LIMIT 50000

Results (731)

PART_DT	GMV	EXPR$2
2012-01-03	917.4138	1
2012-01-04	553.0541	2
2012-01-01	466.9037	3
2012-01-02	970.2347	4
2012-01-16	464.9043	5
2012-01-15	459.9032	6
2012-01-14	625.2434	7
2012-01-13	953.0825	8
2012-01-20	1292.9874	9

Q & A