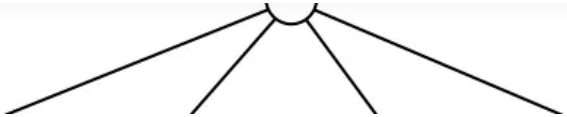
Kylin 维度高级设置

Alex90 关注

0.184 2019.08.29 18:13:05 字数 2,026 阅读 194

> 来源公众号：apachekylin

Apache Kylin 的主要工作就是为源数据构建 N 个维度的 Cube，实现聚合的预计算。理论上而言，构建 N 个维度的 Cube 会生成 2^N 个 Cuboid， 如图所示，构建一个 4 个维度（A，B，C，D）的 Cube，需要生成 16 个 Cuboid。



cube

随着维度数目的增加 Cuboid 的数量会爆炸式地增长，不仅占用大量的存储空间还会延长 Cube 的构建时间。为了缓解 Cube 的构建压力，减少生成的 Cuboid 数目，占用存储空间，同时提高查询性能，Apache Kylin 引入了一系列的高级设置，帮助用户筛选出真正需要的 Cuboid。这些高级设置包括 聚合组（Aggregation Group） 、 联合维度（Joint Dimension） 、 层级维度（Hierachy Dimension） 和 必要维度（Mandatory Dimension） 等。

聚合组

> 聚合组适用于粗粒度地关注某些维度去进行分组聚合的场景。

用户根据自己关注的维度组合，可以划分出自己关注的组合大类，这些大类在 Apache Kylin 里面被称为聚合组。上面的例子如果用户仅仅关注维度 AB 组合和维度 CD 组合，那么该 Cube 则可以被分化成两个聚合组，分别是聚合组 AB 和聚合组 CD。如图所示，生成的 Cuboid 数目从 16 个缩减成了 8 个。

推荐阅读

佟丽娅穿"垃圾花裙"上热搜：你一定想不到，这些时髦单品其实是垃圾...
阅读 10,916

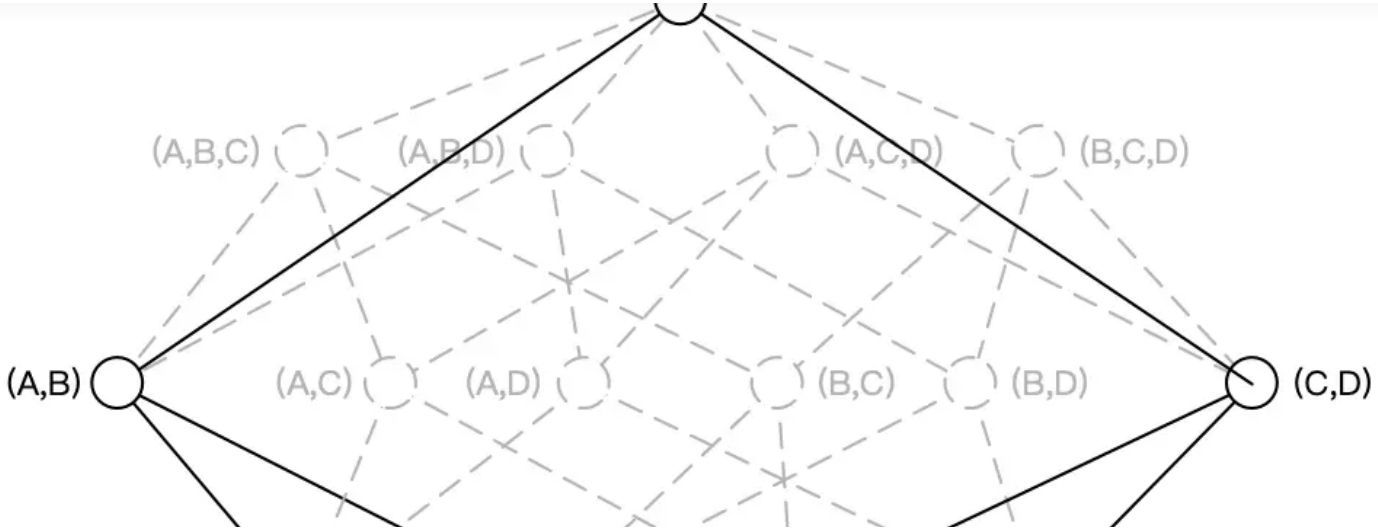
中国最毁三观的节目，把爱情以保卫的名义拿出来暴晒，竟十年长红
阅读 9,111

复旦女博士脚踏3男：为什么偷走爱情的女人，总是相貌平平?
阅读 36,010

卖不掉的房子，被套在北京的我们
阅读 50,475

21世纪顶级恐怖片全在这
阅读 9,783

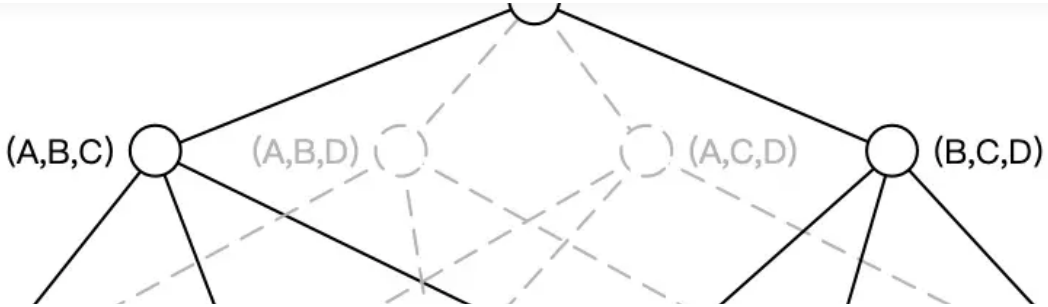
Alex90 关注 赞赏支持



aggregation

用户关心的聚合组之间可能包含相同的维度，例如聚合组 ABC 和聚合组 BCD 都包含维度 B 和维度 C。这些聚合组之间会衍生出相同的 Cuboid，例如聚合组 ABC 会产生 Cuboid BC，聚合组 BCD 也会产生 Cuboid BC。这些 Cuboid不会被重复生成，一份 Cuboid 为这些聚合组所共有，如图所示。

Alex90 关注 赞赏支持



aggregation

有了聚合组用户就可以粗粒度地对 Cuboid 进行筛选，获取自己想要的维度组合。

应用实例

假设创建一个交易数据的 Cube，包含了以下一些维度：顾客 ID（buyer_id）、交易日期（cal_dt）、付款方式（pay_type）和买家所在的城市（city）。分析师有时候需要通过分组聚合 city 、cal_dt 和 pay_type 来获知不同消费方式在不同城市的应用情况；有时候，需要通过聚合 city 、cal_dt 和 buyer_id，来查看顾客在不同城市的消费行为。

在上述的实例中，推荐建立两个聚合组：
聚合组 1： [cal_dt, city, pay_type]
聚合组 2： [cal_dt, city, buyer_id]

在不考虑其他干扰因素的情况下，这样的聚合组将节省不必要的 3 个 Cuboid（参考上图）：[pay_type, buyer_id]、[city, pay_type, buyer_id] 和 [cal_dt, pay_type, buyer_id]，节省了存储资源和构建的执行时间。

Case 1：

将从 Cuboid [cal_dt, city, pay_type] 中获取数据。

Case2：

```
SELECT cal_dt, city, buy_id, count(*) FROM table GROUP BY cal_dt, city, buyer_id;
```

将从 Cuboid [cal_dt, city, pay_type] 中获取数据。

Case3：
如果有一条不常用的查询

```
SELECT pay_type, buyer_id, count(*) FROM table GROUP BY pay_type, buyer_id;
```

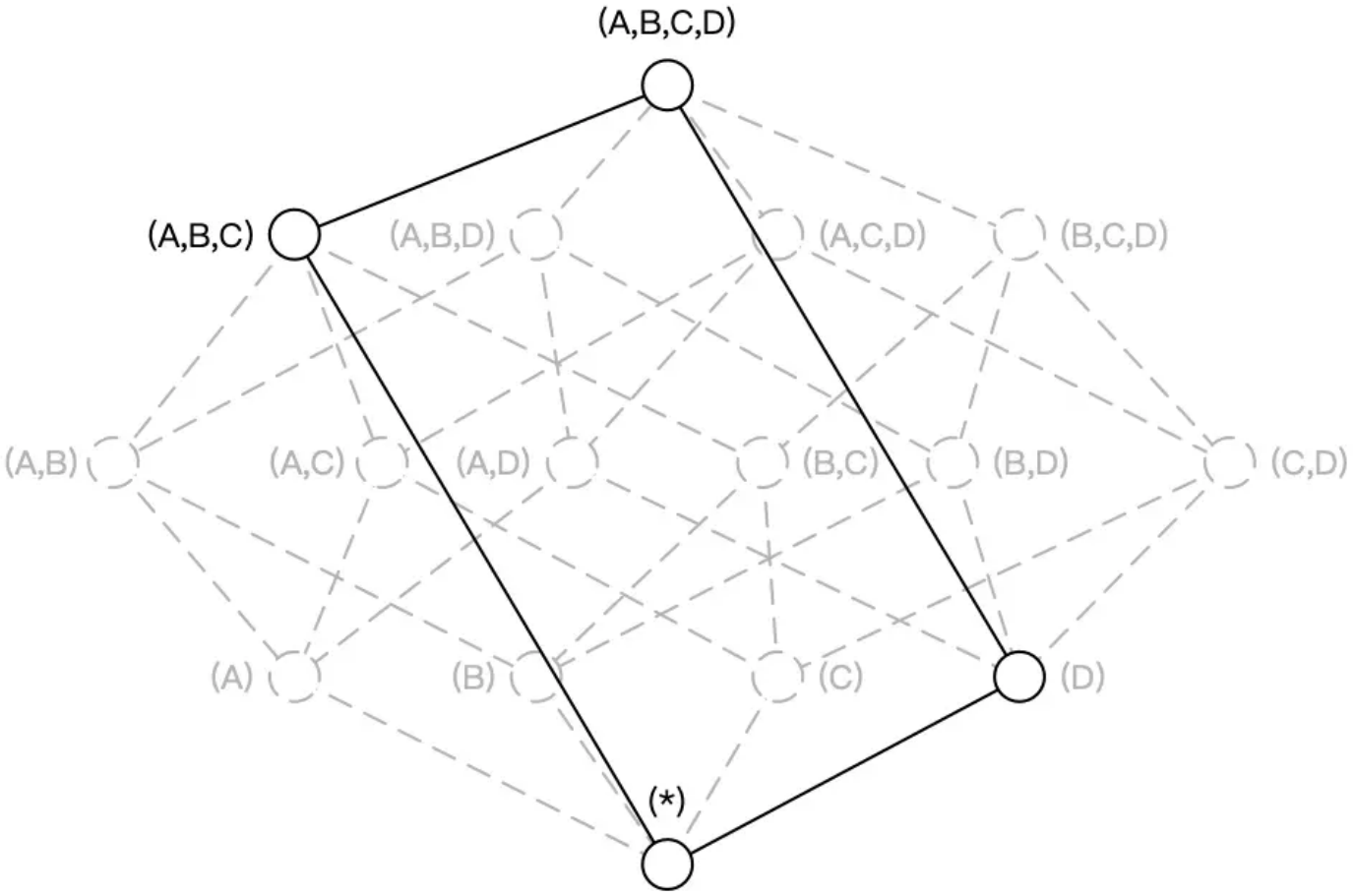
不存在匹配的 Cuboid [pay_type, buyer_id]，此时，Apache Kylin 会通过在线计算的方式，从现有的 Cuboid 中计算出最终结果。

联合维度

> 联合维度适用于固定用来分组查询的维度。

用户有时并不关心维度之间各种细节的组合方式，例如用户的查询语句中仅仅会出现 group by A, B, C，而不会出现 group by A, B 或者 group by C 等等这些细化的维度组合。这一类问题就是联合维度所解决的问题。

4。



joint

应用实例

假设创建一个交易数据的 Cube，包含了以下一些维度：交易日期（cal_dt）、交易城市

Alex90 关注 赞赏支持

在上述的实例中，推荐在已有的聚合组中建立一组联合维度：

聚合组：[cal_dt, city, sex_id，pay_type]

联合维度： [cal_dt, city, sex_id]

在不考虑其他干扰因素的情况下，将创建 4 个 Cuboid（参考上图三）： [cal_dt, city, sex_id, pay_type]、[cal_dt, city, sex_id]、[pay_type] 和 [*]，节省了存储资源和构建的执行时间。

Case 1：

```
SELECT cal_dt, city, sex_id, count(*) FROM table GROUP BY cal_dt, city, sex_id;
```

从Cuboid [cal_dt, city, sex_id]中获取数据

Case2：

如果有一条不常用的查询

```
SELECT cal_dt, city, count(*) FROM table GROUP BY cal_dt, city;
```

不存在匹配的 Cuboid [cal_dt, city]。此时，Apache Kylin 会通过在线计算的方式，从现有的 Cuboid 中计算出最终结果。

层级维度

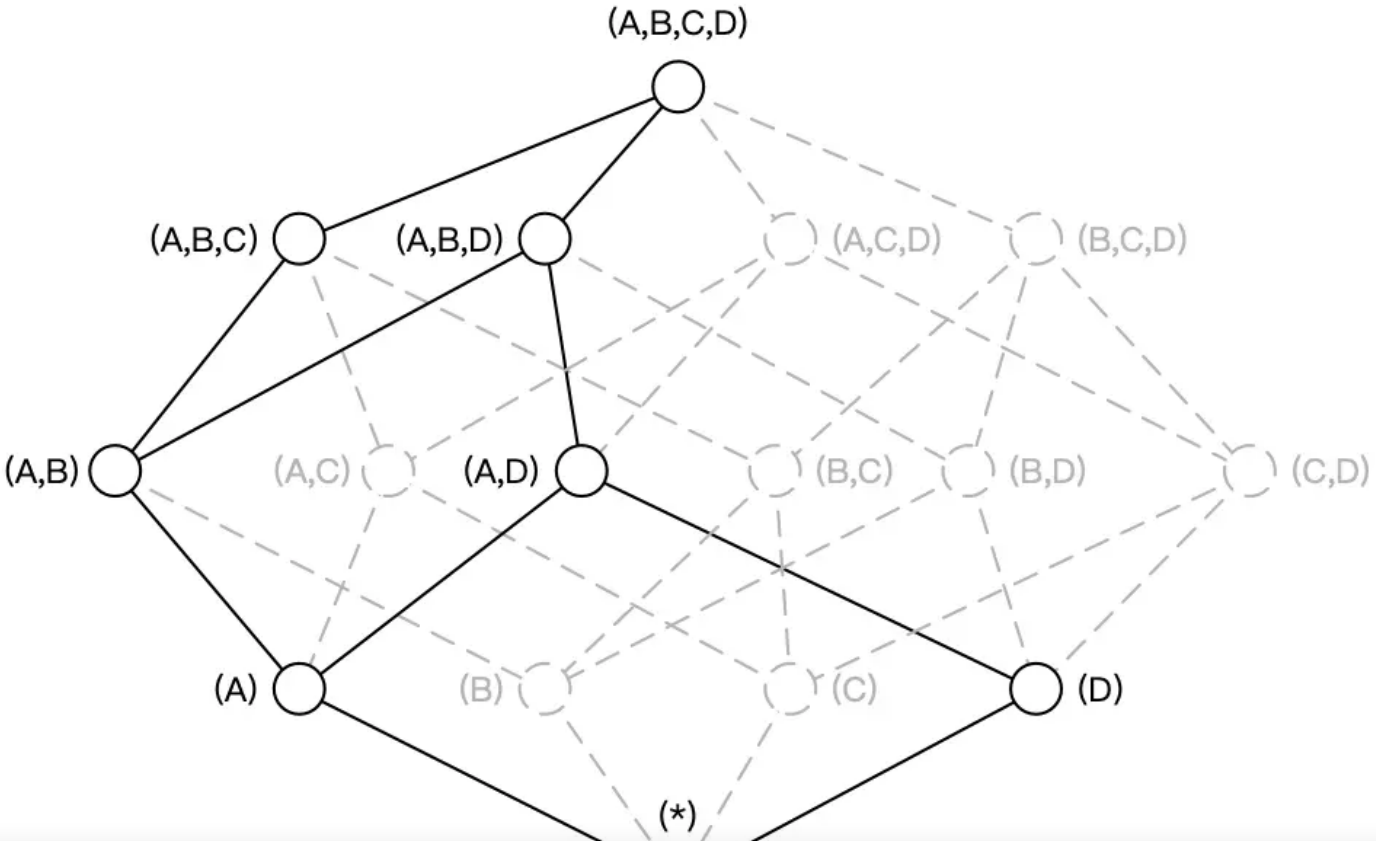
> 层级维适用于维度间有一对多关系的场景，比如国家 / 省 / 城市，产品大类 / 产品子类等

Alex90 关注 赞赏支持

系。也就是说，用户对于这三个维度的查询可以归类为以下三类:

- group by country
- group by country, province（等同于 group by province）
- group by country, province, city（等同于 group by city）

基于ABCD四个维度的场景，假设维度 A 代表国家，维度 B 代表省份，维度 C 代表城市，那么 ABC 三个维度可以被设置为层级维度，生成的Cube 如图所示，Cuboid 数目从 16 减小到 8。



推荐阅读

佟丽娅穿"垃圾花裙"上热搜：你一定想不到，这些时髦单品其实是垃圾...
阅读 10,916

中国最毁三观的节目，把爱情以保卫的名义拿出来暴晒，竟十年长红
阅读 9,111

复旦女博士脚踏3男：为什么偷走爱情的女人，总是相貌平平?
阅读 36,010

卖不掉的房子，被套在北京的我们
阅读 50,475

21世纪顶级恐怖片全在这
阅读 9,783

Alex90 关注 赞赏支持

Cuboid [A,C,D]=Cuboid[A, B, C, D]，Cuboid[B, D]=Cuboid[A, B, D]，因而 Cuboid[A, C, D] 和 Cuboid[B, D] 就不必重复存储。

应用实例

假设创建一个交易数据的 Cube，包含了以下一些维度：交易城市（city），交易省（province），交易国家（country）和支付类型（pay_type）等。分析师可以通过按照交易城市、交易省份、交易国家和支付类型来聚合，获取不同层级的地理位置消费者的支付偏好。

在上述的实例中，建议在已有的聚合组中建立一组层级维度（国家country / 省province / 城市city），包含的维度和组合方式：
聚合组：[country, province, city，pay_type]
层级维度： [country, province, city]

Case 1：
从城市维度获取消费偏好

```
SELECT city, pay_type, count(*) FROM table GROUP BY city, pay_type;
```

将从 Cuboid [country, province, city, pay_type] 中获取数据。

Case 2：
从省级维度获取消费偏好

```
SELECT province, pay_type, count(*) FROM table GROUP BY province, pay_type;
```

Alex90 关注 赞赏支持

Case 3：

从国家维度获取消费偏好

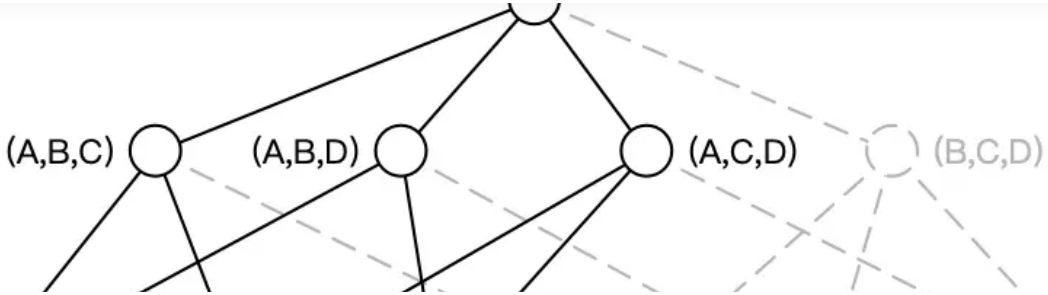
```
SELECT country, pay_type, count(*) FROM table GROUP BY country, pay_type;
```

将从Cuboid [country, pay_type] 中获取数据。

必要维度

> 必要维度适用于某些维度被高频使用的情景下

用户有时会对某一个或几个维度特别感兴趣，所有的查询请求中都存在 group by 这个维度，那么这个维度就被称为必要维度，只有包含此维度的 Cuboid 会被生成。假设维度A是必要维度，那么生成的 Cube 如图所示，维度数目从16变为9。



mandatory

应用实例

假设创建一个交易数据的 Cube，包含了以下一些维度：交易时间（order_dt）、交易地点（location）、交易商品（product）和支付类型（pay_type）等。其中，交易时间就是一个被高频作为分组条件（group by）的维度。 如果将交易时间 order_dt 设置为必要维度，只有带有 order_dt 的 Cuboid 会被创建