> 关注公众号：大数据技术派，回复"资料"，领取`1024G`资料。

宽表的设计

其实宽表是数仓里面非常重要的一块，前面我们介绍过了维度表事实表，今天我们介绍一下宽表，前面我们说过了数仓是分层的，这是技术进步和时代变化相结合的产物，数仓的分层式为了更好地管理数仓以及更加高效地进行数据开发。

宽表主要出现在dwd 层和报表层，当然有的人说dws 层也有，宽表，从字面意义上讲就是字段比较多的数据库表，通常情况下是将很多相关的数据包括维度表、实时、已有的指标或者是dws/dwd 表关联在一起形成的一张数据表。

由于把不同的内容都放在同一张表存储，宽表已经不符合范式设计的模型设计规范而且数仓里面也不强调范式设计，随之带来的就是数据的大量冗余，与之相对应的好处就是查询性能的提高与便捷。

分层 请参考 数仓建模—分层建设理论

设计 请参考 数仓建模—建模方法论

为什么要建设宽表

就像我们前面说过分层的目的是为了管理方便、开发高效、问题定位、节约资源等等，那么我们建设宽表呢？前面学习建模方法论的时候，提到过维度模型的非强范式的，**可以更好的利用大数据处理框架的处理能力，避免范式操作的过多关联操作，可以实现高度的并行化。**数据仓库大多数时候是比较适合使用星型模型构建底层数据Hive表，通过大量的冗余来提升查询效率，星型模型对OLAP的分析引擎支持比较友好，这一点在Kylin中比较能体现。

可以更好的发挥大数据框架的能力

维度模型可以更好地利用大数据框架，体现在哪里的，体现在数据数据冗余，可以避免很多的关联，怎么体现的呢，宽表。但是这只是站在大数据框架层面上的理解，还有其他层面上的理解。

可以提高开发效率

一般情况下，我们的宽表包含了很多相关的数据，如果我们在宽表的基础上做一些开发，那就很方便，我们直接从宽表里面取数据，避免了我们从头计算，你设想一下你要是没次都从ods开发一张报表，那是多痛苦的体验啊。

可以提高数据质量

宽表的准确性，一般都是经历了时间的检验的，逻辑错误的可能性很小，可以直接使用，要是让你从头开发，那这个过程中可能因为对业务理解不透彻或者是书写的逻辑不正确，导致有数据质量问题

可以统一指标口径

其实这一点和上面一点有点重复，但是这两点的强调的方面是不一样的，因为如果我们的报表要是都能从我们的底层宽表出，那么我们报表上的指标肯定是一样的，其实这一点我相信很多人都深有体会，同一个指标的口径不一致，导致我们提供的数据在不同的出口不一样，是业务部门经常提出的一个问题。其实这也就是我们一直强调的核心逻辑下沉的原因。

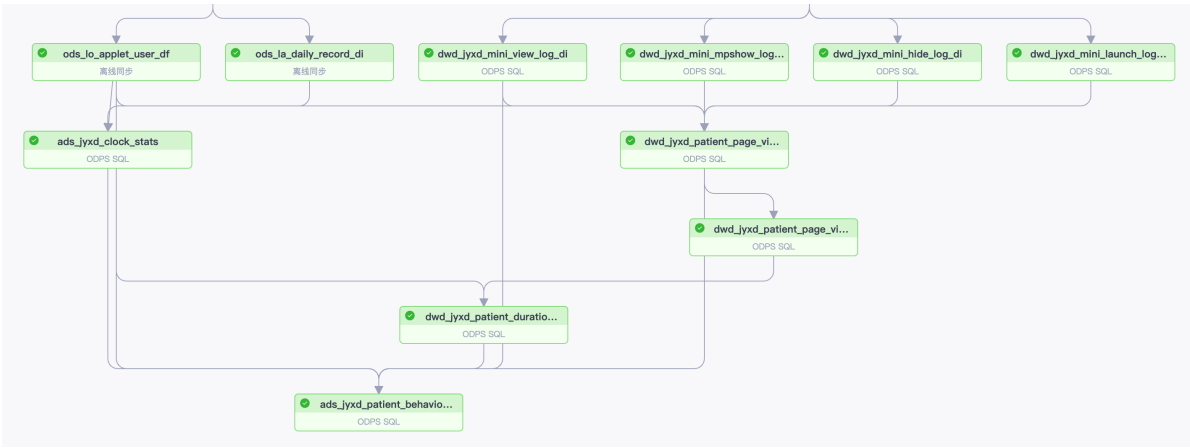
宽表的好处和不足

宽表的好处就是我们前面提到过的我们为什么要建设宽表的原因，接下来我们看一下宽表的不足

性能不高

因为我们的宽表的计算逻辑往往很复杂，再加上宽表的数据输入是有大量依赖的，也就是说需要处理的数据量很大，在负载逻辑+大数据量的原因下，导致我们的宽表往往运行很慢，资源占用很多，尤其是重跑的时候。

稳定性不高



下面的最后一张表就是一张宽表，我们知道一个系统的稳定性是取决于最差的一个环节的，这就是短板理论也叫木桶理论，我们的宽表的稳定性也是很差的，这个主要是因为我们的宽表依赖太多，每一个表的不稳定性都会传到到宽表。

假设 一张表依赖A B C 三张表，并且这三张表的稳定性是 1/m 1/n 1/x,那么我们的宽表的稳定性就是 1/m*n*x ,至于表的稳定性你可用它成功运行的次数/运行的总次数

如果性能不高和稳定性不高同时作用在一件事上的时候我们知道这其实是很致命的，例如你发现报表数据有问题，但是重跑需要几个小时，哈哈！

开发难度大/维护成本高

我们说了基于宽表做报表开发才是正确的姿势，但是宽表本身也是我们开发人员开发的，因为本身的逻辑很复杂设计的业务逻辑繁多，所以给我们的开发就带来了挑战，而且由于业务逻辑的变更我们也需要去维护着复杂的逻辑，例如每次都让你在几千行的SQL 里面加逻辑。

如何设计宽表

宽表的好处和不足我们都讲了，也就是说宽表虽好，但是带来的问题也很多，下面我们就看一下如何从设计的角度来避免宽表的不足之处

宽表到底多宽

开始之前，我们思考一个问题，那就是宽表到底有多宽，就想我们前面讲分层的时候说其实我们不分层也玩得转，早起的数仓就只有一层，现在我们考虑一个问题那就是宽表到底多宽才合适，其实你要把所有的数据装进去也可以。

所以我们要思考到底多宽才合适的，前面我们介绍过数据域的概念，我们与其回答多宽这个问题，不如回答宽表都应该覆盖哪些数据，但是这个问题也不好回答，但是我们可以反着思考，宽表不应该包含什么数据，这个问题很好回答，宽表不应该包含不属于它所在域的数据，例如会员域的宽表只应该包含会员相关的信息，同理我们的宽表是针对某一个域而言的，也就是说它是有边界的。

这下我们再来回答宽表到底多宽，只要不跨域，并且方便使用都是合理的。但是这似乎并不能解决我们上面提到的宽表的不足，只是指明了宽表的一个大致的方向。有了方向之后我们通过我们的设计策略就可以让宽表瘦下来。

主次分类

主次分离，其实我们经常听到的一句话就是做事情要搞清楚主次，我们看一下表设计的主次是什么，假设我们做的是一个会员域的宽表，但是会员域是还是一个比较大的概念，所以我们还要发掘出我们这个表的主题，例如我们做的是一张会员域下的会员基本信息宽表，那么我们专注的肯定就是基本信息，例如会员信息打通。当让因为事宽表你可能还会冗余的其他信息进来，但是当这样的信息越来越多的时候，我们这张表的主题就越来越弱，所以我们就需要做拆分。

拆分可以让我们更加聚焦表的主题，对于数仓开发人员而言可以更好的维护、对于使用方而言可以更加清楚的理解这张表的主题。

冷热分离

除了前面的主次分离我们还可以做冷热分离，其实冷热分离这个词我相信你不是第一次听到，但是怎么看这个事情呢，你想一下你在数据存储的时候是怎么做冷热分离的，这里也是同样的理念。

假设我有一张宽表，里面有200个字段，有30张报表在使用它，但是我发现前面150个经常字段经常被使用，后面 50个字段只有一两张报表使用到了，那么我们就可以做一个冷热分离，将宽表拆分。

稳定与不稳定分离

其实前面的主次分离、冷热分离都可以提高稳定性，但是前面我们不是为了稳定性分离的。

我们经常有这样的宽表，它依赖埋点数据，但是我们的埋点数据的特点就是量大，导致计算经常延迟，那么我们的宽表就会受影响，从而我们的报表就受影响，但是很多时候你发现报表根本没有用过埋点计算出来的指标，或者是只用了一两个。那我们可以将其拆分，如果报表没有使用到那就最好了，如果使用到了，那就后推，在报表层面上做关联，这样我们的埋点数据即使出不来，我们的报表数据还是可以看的。

总结

主要讲解了一下几个方面

1. 为什么要建设宽表
2. 宽表的不足
3. 如何设计宽表
    1. 宽表到底多宽
    2. 主次分离
    3. 冷热分类
    4. 稳定与不稳定分类

设计宽表的理论其实说白了就是一句话`高内聚低耦合`，当然这几个字你在其他领域可能很熟悉了，但是这里你就好好思考一下才能想通，我一直新信奉的是`一力降十会 一拙破万巧`也就是说你要学会根本的东西，才能举一反三破万难。

猜你喜欢

Spark SQL知识点与实战

Hive计算最大连续登陆天数

Hadoop 数据迁移用法详解

Hbase修复工具Hbck

数仓建模分层理论

Flink 是如何统一批流引擎的