尚硅谷大数据之电商实时数仓
之
实时数仓设计以及开发环境准备

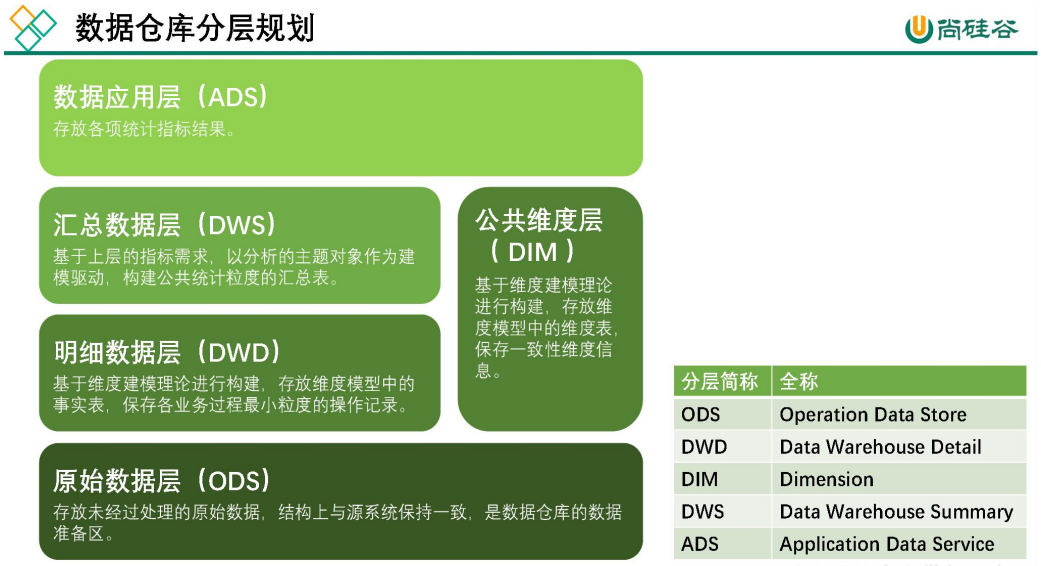
（作者：尚硅谷研究院）

版本：V3.0

第 1 章　数据仓库设计

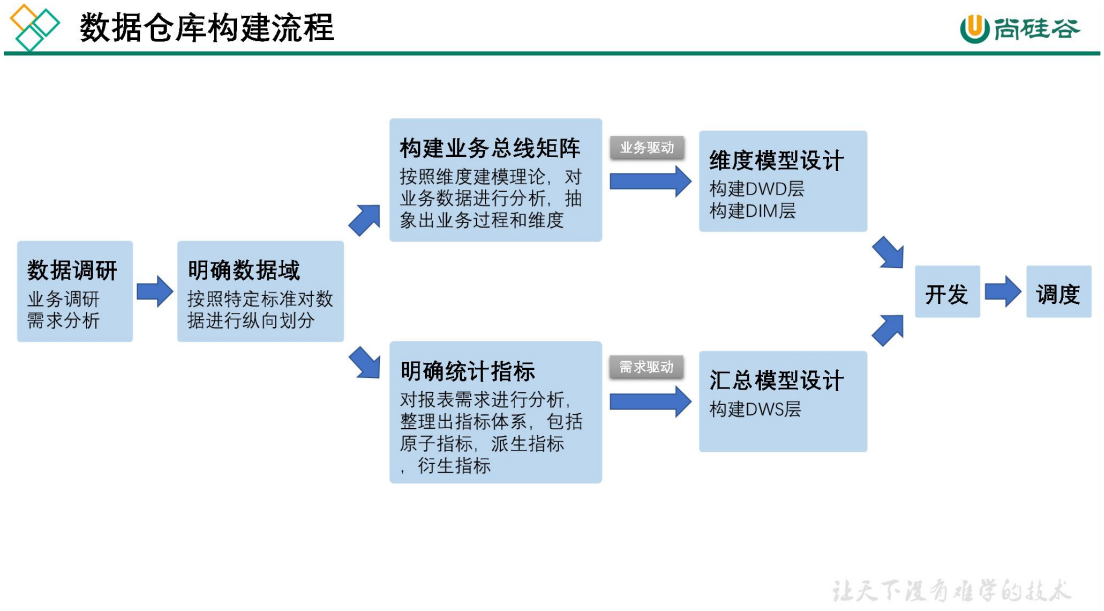
1.1　数据仓库分层规划

优秀可靠的数仓体系，需要良好的数据分层结构。合理的分层，能够使数据体系更加清晰，使复杂问题得以简化。以下是该项目的分层规划。



1.2　数据仓库构建流程

以下是构建数据仓库的完整流程。



1.2.1 数据调研

数据调研重点要做两项工作，分别是业务调研和需求分析。这两项工作做的是否充分，直接影响着数据仓库的质量。

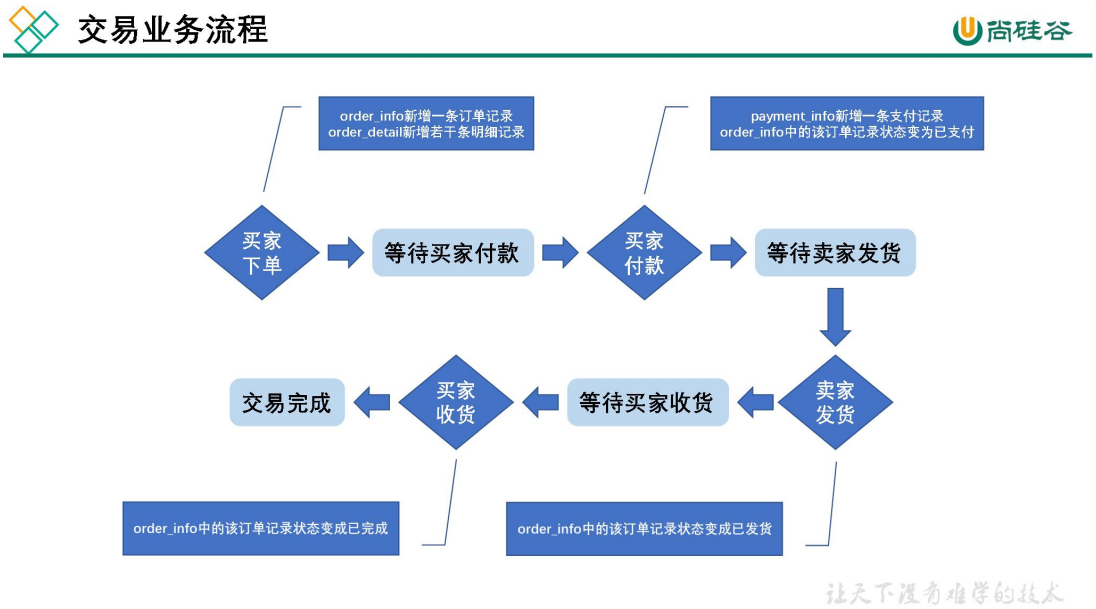
(1) 业务调研

业务调研的主要目标是**熟悉业务流程**、**熟悉业务数据**。

熟悉业务流程要求做到，明确每个业务的具体流程，需要将该业务所包含的每个**业务过程**一一列举出来。

熟悉业务数据要求做到，将数据（包括埋点日志和业务数据表）与业务过程对应起来，明确每个业务过程会对哪些表的数据产生影响，以及产生什么影响。产生的影响，需要具体到，是新增一条数据，还是修改一条数据，并且需要明确新增的内容或者是修改的逻辑。

下面业务电商中的交易为例进行演示，交易业务涉及到的业务过程有买家下单、买家支付、卖家发货，买家收货，具体流程如下图。



(2) 需求分析

典型的需求指标如，最近一天各省份手机品类订单总额。

分析需求时，需要明确需求所需的**业务过程**及**维度**，例如该需求所需的业务过程就是买

家下单，所需的维度有日期，省份，商品品类。

(3) 总结

做完业务分析和需求分析之后，要保证每个需求都能找到与之对应的业务过程及维度。若现有数据无法满足需求，则需要和业务方进行沟通，例如某个页面需要新增某个行为的埋点。

1.2.2 明确数据域

数据仓库模型设计除横向的分层外，通常也需要根据业务情况进行纵向划分数据域。划分数据域的意义是**便于数据的管理和应用**。

通常可以根据业务过程或者部门进行划分，本项目根据业务过程进行划分，需要注意的是一个业务过程只能属于一个数据域。

下面是本数仓项目所需的所有业务过程及数据域划分详情。

数据域	业务过程
交易域	加购、下单、取消订单、支付成功、退单、退款成功
流量域	页面浏览、启动应用、动作、曝光、错误
用户域	注册、登录
互动域	收藏、评价
工具域	优惠券领取、优惠券使用（下单）、优惠券使用（支付）

1.2.3 构建业务总线矩阵

业务总线矩阵中包含维度模型所需的所有事实（业务过程）以及维度，以及各业务过程与各维度的关系。矩阵的行是一个个业务过程，矩阵的列是一个个的维度，行列的交点表示业务过程与维度的关系。

一个业务过程对应维度模型中一张事务型事实表，一个维度则对应维度模型中的一张维度表。所以构建业务总线矩阵的过程就是设计维度模型的过程。但是需要注意的是，总线矩阵中通常只包含事务型事实表，另外两种类型的事实表需单独设计。

按照事务型事实表的设计流程，**选择业务过程→声明粒度→确认维度→确认事实**，得到的最终的业务总线矩阵见以下表格。

业务总线矩阵.xlsx

后续的 DWD 层以及 DIM 层的搭建需参考业务总线矩阵。

1.2.4 明确统计指标

明确统计指标具体的工作是，深入分析需求，构建指标体系。构建指标体系的主要意义就是指标定义标准化。所有指标的定义，都必须遵循同一套标准，这样能有效的避免指标定义存在歧义，指标定义重复等问题。

(1) 指标体系相关概念

- **原子指标**

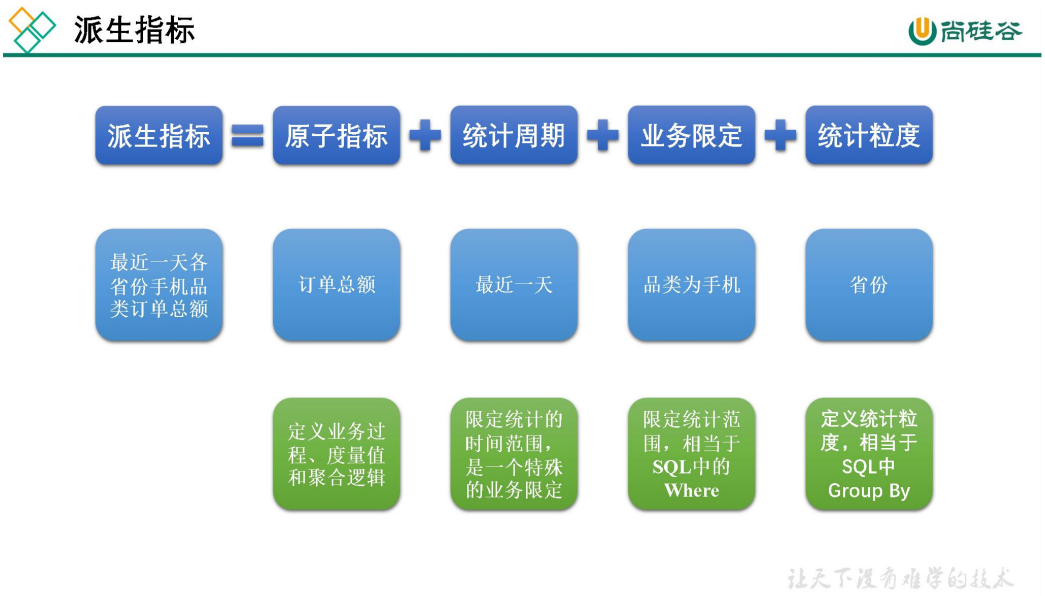
原子指标基于某一**业务过程**的**度量值**，是业务定义中不可再拆解的指标，原子指标的核

心功能就是对指标的**聚合逻辑**进行了定义。我们可以得出结论，原子指标包含三要素，分别是业务过程、度量值和聚合逻辑。

例如**订单总额**就是一个典型的原子指标，其中的业务过程为用户下单、度量值为订单金额，聚合逻辑为 sum()求和。需要注意的是原子指标只是用来辅助定义指标一个概念，通常不会对应有实际统计需求与之对应。

- **派生指标**

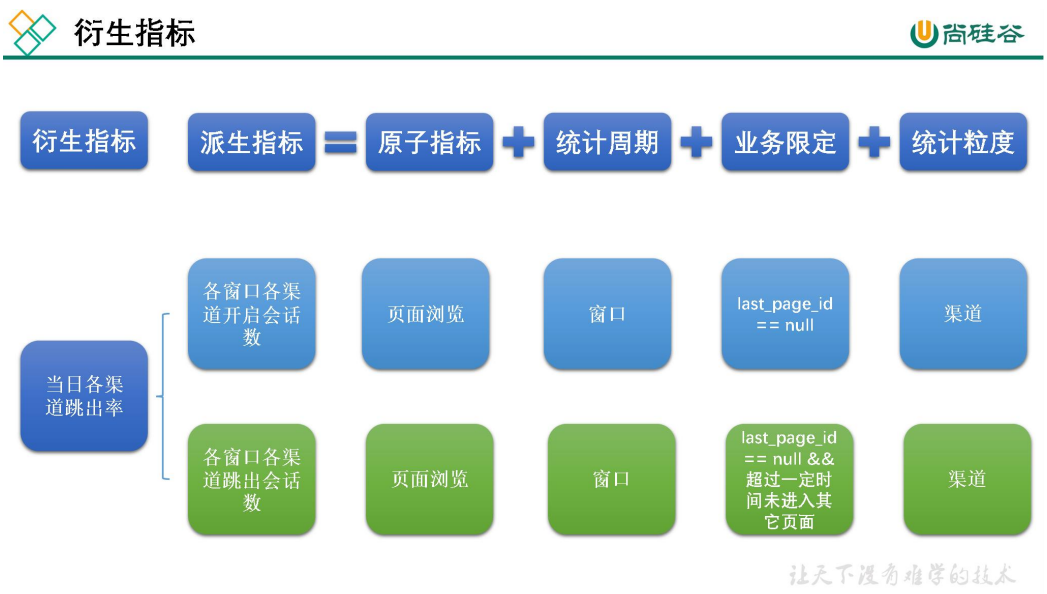
派生指标基于原子指标，其与原子指标的关系如下图所示。



与原子指标不同，派生指标通常会对应实际的统计需求。请从图中的例子中，体会指标定义标准化的含义。

- **衍生指标**

衍生指标是在一个或多个派生指标的基础上，通过各种逻辑运算复合而成的。例如比率、比例等类型的指标。衍生指标也会对应实际的统计需求。



(2) 指标体系对于数仓建模的意义

通过上述两个具体的案例可以看出，绝大多数的统计需求，都可以使用原子指标、派生指标以及衍生指标这套标准去定义。同时能够发现这些统计需求都直接的或间接的对应一个或者是多个派生指标。

当统计需求足够多时，必然会出现部分统计需求对应的派生指标相同的情况。这种情况下，我们就可以考虑将这些公共的派生指标保存下来，这样做的主要目的就是减少重复计算，提高数据的复用性。

这些公共的派生指标统一保存在数据仓库的 DWS 层。因此 DWS 层设计，就可以参考我们根据现有的统计需求整理出的派生指标。

从上述指标体系中抽取出来的所有派生指标见如下表格。

gmall-实时重构指标体系.xlsx

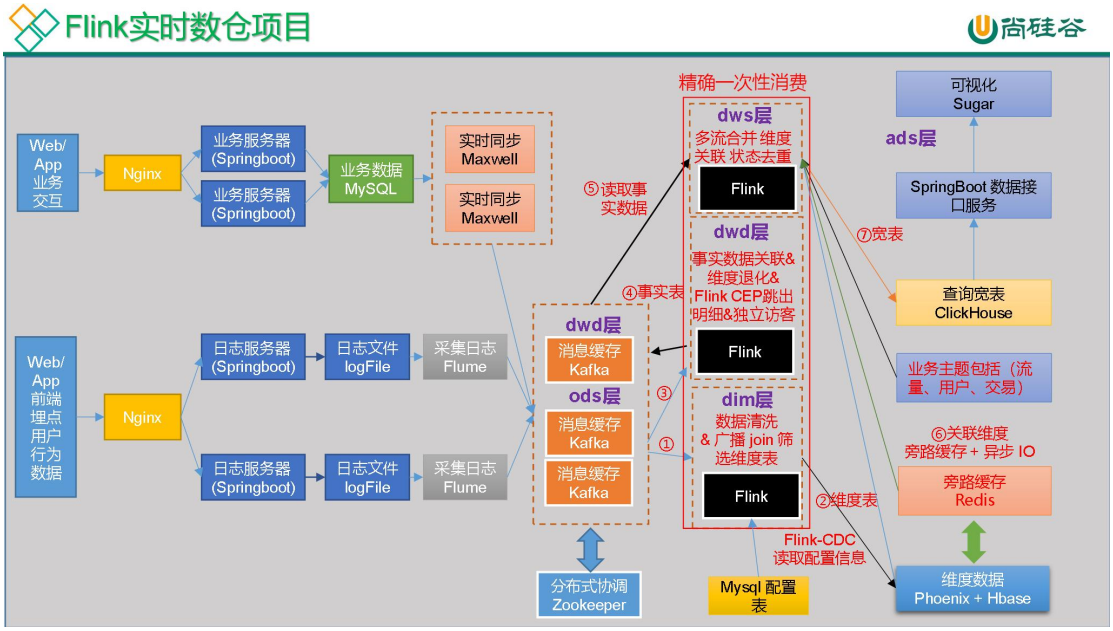
1.2.5 维度模型设计

维度模型的设计参照上述得到的业务总线矩阵即可。事实表存储在 DWD 层，维度表存储在 DIM 层。

1.2.6 汇总模型设计

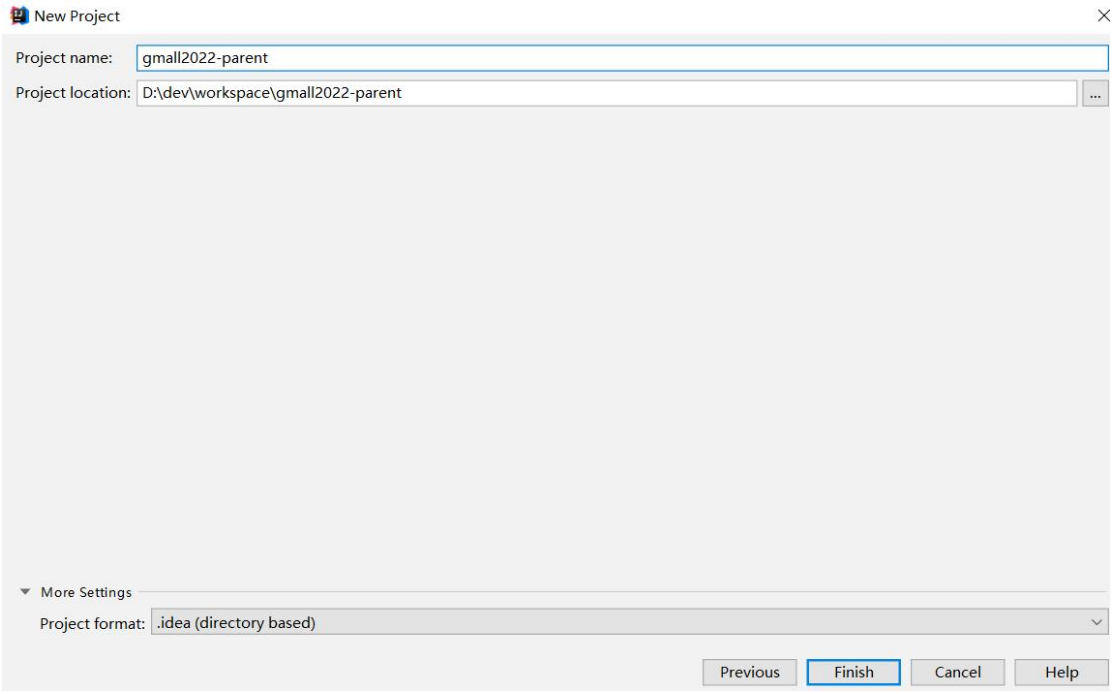
汇总模型的设计参考上述整理出的指标体系（主要是派生指标）即可。汇总表与派生指标的对应关系是，**一张汇总**表通常**包含**业务过程相同、统计周期相同、统计粒度相同的**多个派生指标**。

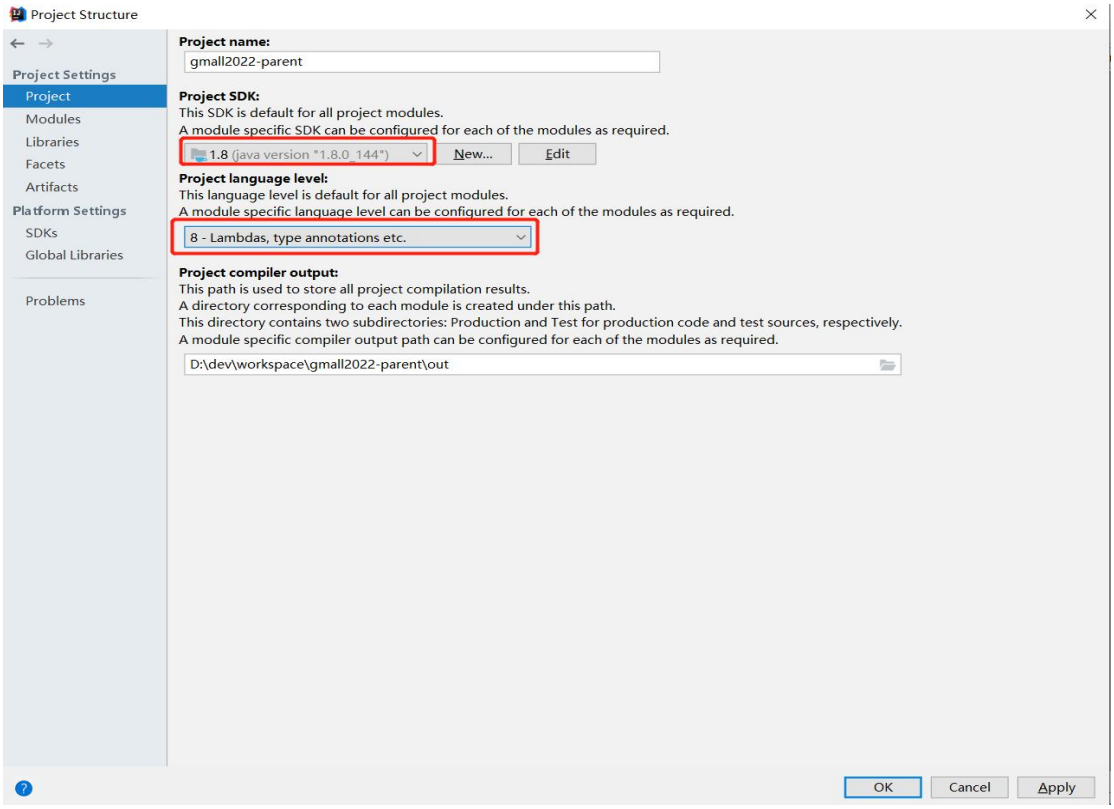
第 2 章 实时数仓架构图



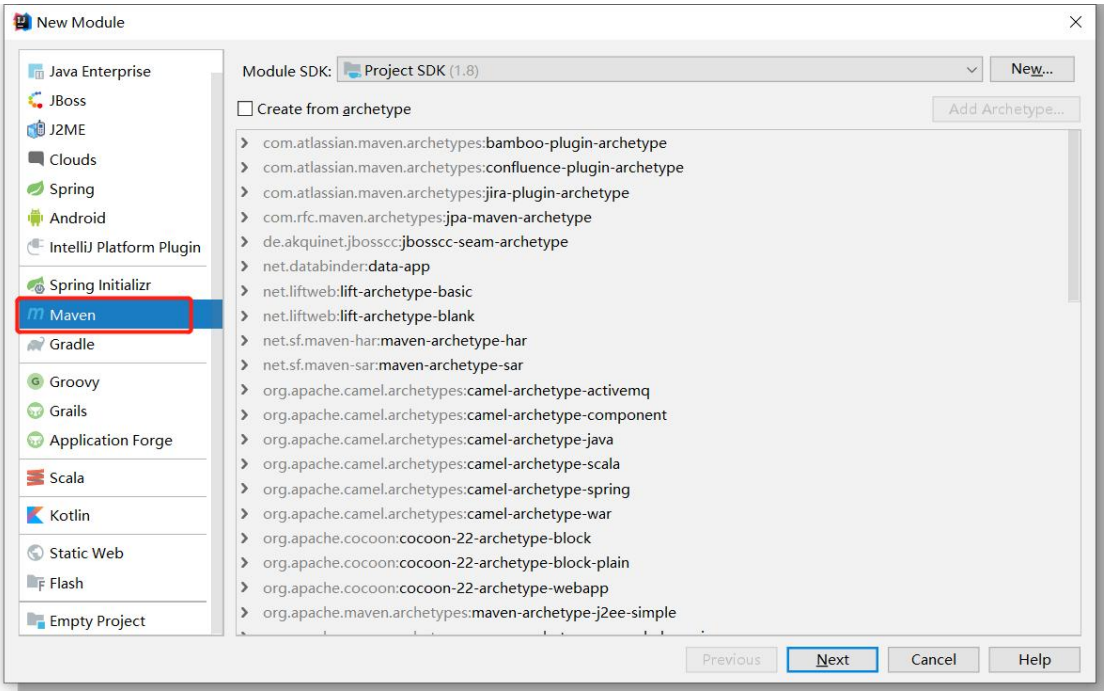
第 3 章　IDEA 开发环境准备

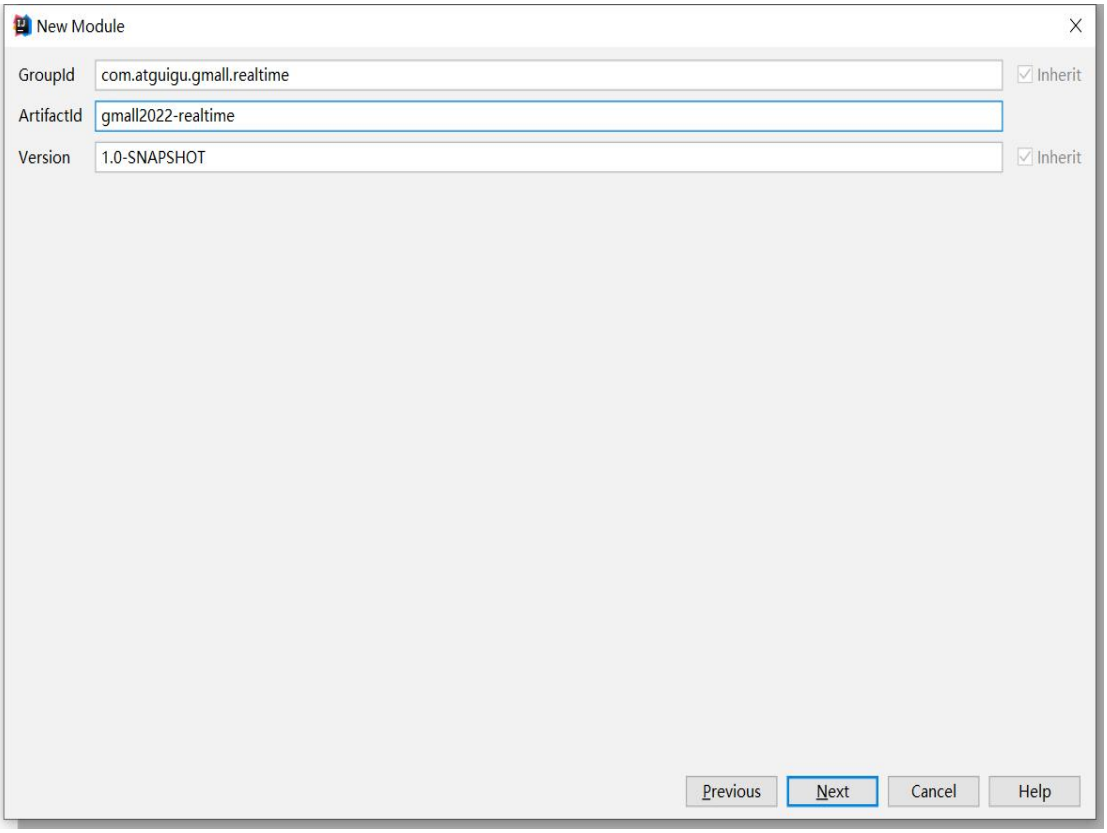
3.1　创建空的 project 项目 gmall2022-parent

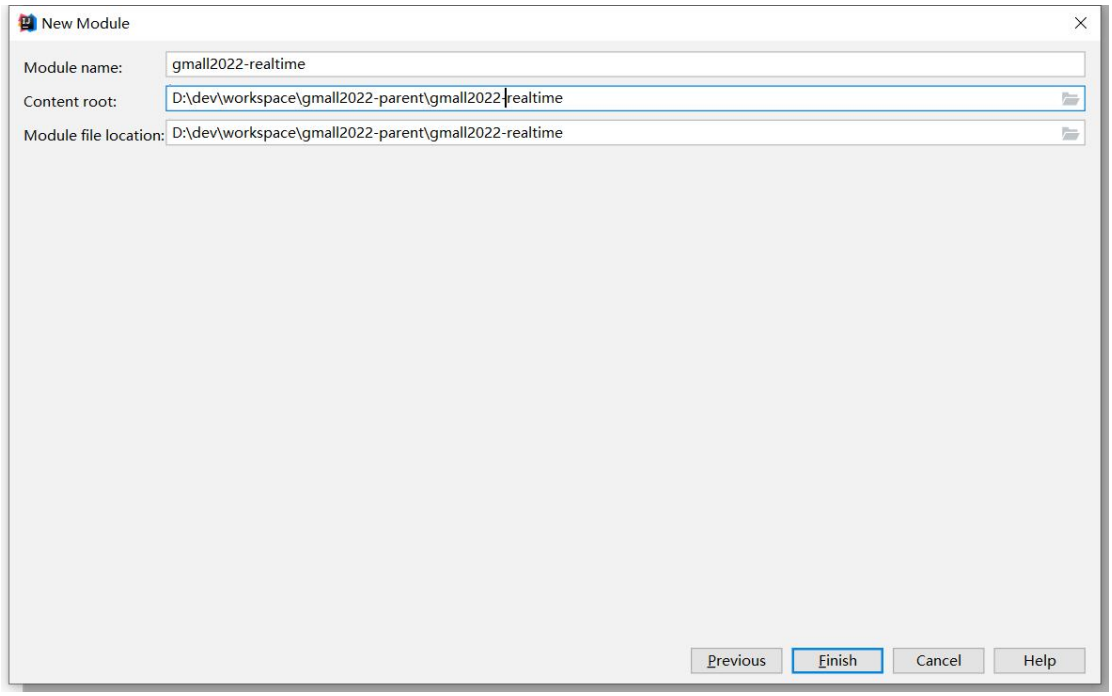




3.2 在当前 project 下创建 gmall2022-realtime 模块







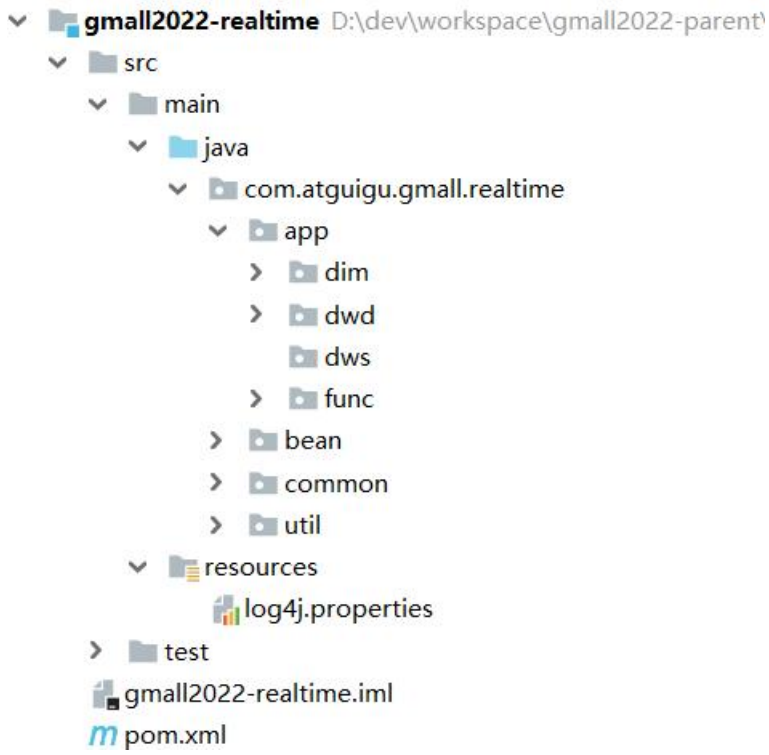
3.3 导入依赖

```
<properties>
    <java.version>1.8</java.version>
    <maven.compiler.source>${java.version}</maven.compiler.source>
    <maven.compiler.target>${java.version}</maven.compiler.target>
    <flink.version>1.13.0</flink.version>
    <scala.version>2.12</scala.version>
    <hadoop.version>3.1.3</hadoop.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-java</artifactId>
        <version>${flink.version}</version>
    </dependency>

    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-streaming-java_${scala.version}</artifactId>
        <version>${flink.version}</version>
    </dependency>

    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-connector-kafka_${scala.version}</artifactId>
        <version>${flink.version}</version>
    </dependency>

    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-clients_${scala.version}</artifactId>
        <version>${flink.version}</version>
    </dependency>
```

```
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-json</artifactId>
        <version>${flink.version}</version>
    </dependency>

    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>fastjson</artifactId>
        <version>1.2.68</version>
    </dependency>

    <!--如果保存检查点到 hdfs 上，需要引入此依赖-->
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-client</artifactId>
        <version>${hadoop.version}</version>
    </dependency>

    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.20</version>
    </dependency>

    <!--Flink 默认使用的是 slf4j 记录日志，相当于一个日志的接口,我们这里使用 log4j 作为具体
的日志实现-->
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>1.7.25</version>
    </dependency>

    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
        <version>1.7.25</version>
    </dependency>

    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-to-slf4j</artifactId>
        <version>2.14.0</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
            <version>3.1.1</version>
            <executions>
                <execution>
                    <phase>package</phase>
                    <goals>
                        <goal>shade</goal>
                    </goals>
                    <configuration>
                        <artifactSet>
                            <excludes>
                                <exclude>com.google.code.findbugs:jsr305</exclude>
```

```
                            <exclude>org.slf4j:*</exclude>
                            <exclude>log4j:*</exclude>
                            <exclude>org.apache.hadoop:*</exclude>
                        </excludes>
                    </artifactSet>
                    <filters>
                        <filter>
                            <!-- Do not copy the signatures in the META-INF
folder.Otherwise, this might cause SecurityExceptions when using the JAR. -->
                            <!-- 打包时不复制 META-INF 下的签名文件，避免报非法签名文件
的 SecurityExceptions 异常-->
                            <artifact>*:*</artifact>
                            <excludes>
                                <exclude>META-INF/*.SF</exclude>
                                <exclude>META-INF/*.DSA</exclude>
                                <exclude>META-INF/*.RSA</exclude>
                            </excludes>
                        </filter>
                    </filters>

                    <transformers combine.children="append">
                        <!-- The service transformer is needed to merge
META-INF/services files -->
                        <!-- connector 和 format 依赖的工厂类打包时会相互覆盖，需要使
用 ServicesResourceTransformer 解决-->
                        <transformer

implementation="org.apache.maven.plugins.shade.resource.ServicesResourceTran
sformer"/>
                    </transformers>
                </configuration>
            </execution>
        </executions>
    </plugin>
  </plugins>
</build>
```

3.4 创建相关的包



3.5 在 resources 目录下创建 log4j.properties 文件，写入如下内容

```
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd
HH:mm:ss} %10p (%c:%M) - %m%n

log4j.rootLogger=error,stdout
```

第 4 章 数据仓库涉及的组件环境准备

4.1 Flink 环境搭建

Flink 集群部署请参考如下文档。

尚硅谷大数据技术
之Flink.docx

4.2 Hbase 环境搭建

Hbase 集群部署请参考如下文档。

尚硅谷大数据技术
之HBase.docx

4.3 Redis 环境搭建

4.3.1 Redis 安装

Redis 安装请参考如下文档

尚硅谷大数据技术
之 Redis.docx

4.3.2 Redis 配置文件修改

- 切换到 atguigu 用户，拷贝 redis.conf 配置文件，置于 atguigu 家目录下

  若已按照 1 中文档操作，则这一步可以省略

```
[root@hadoop102 atguigu]# su atguigu
[atguigu@hadoop102                redis-6.0.8]$                cp
/opt/module/redis-6.0.8/redis.conf ~/my_redis.conf
```

- 注销 bind 配置项

```
# bind 127.0.0.1
```

- 修改 protected-mode

```
protected-mode no
```

4.4 ClickHouse 环境搭建

ClickHouse 安装请参考如下文档

尚硅谷大数据技术
之 ClickHouse.do