介绍：

数据仓库建设现已成为企业标配，数据仓库面试成为企业招聘考察的重点，现整理36道数仓面试题助力各位同学闯关大数据面试。

关键字：Hive、数据仓库、面试

概述：

数据仓库建设现已成为企业标配，数据仓库面试成为企业招聘考察的重点，现整理36道数仓面试题助力各位同学闯关大数据面试。

### 第1题

Hive与HBase的区别？

参考答案：

1.使用方面的区别

Hive

1)Hive是一个构建在Hadoop平台上的数据仓库。

2)Hive数据存放在HDFS之上。

3)Hive数据查询最终被转化为MapReduce。

HBase

1)HBase是一个基于HDFS平台的NoSQL数据库。

2)HBase数据存储在HDFS之上。

3)HBase是基于数据库本身的实时查询，而非运行MapReduce。

2.特点的区别

Hive

1)Hive方便熟悉SQL的人，快速上手大数据分析。

2)Hive默认计算引擎是MapReduce，所以查询时间比较长。

3)Hive底层计算引擎也可以更换为Spark/Tez。

4)Hive中的表是纯逻辑表，只是表的定义，本身不存储和计算，完全依赖HDFS/MapReduce

HBase

1)HBase本身不支持SQL，需要通过集成Phoenix/Hive才支持SQL。

2)HBase基于一级索引Rowkey进行数据查询，所以查询速度比较快。

3)HBase底层基于Scan进行数据扫描，而非基于MapReduce

4)HBase中的表是物理表，有独立的物理数据结构，查询时可以将数据加载到内存，提升后续的查询效率

3.局限性的区别

Hive

1)Hive目前仅支持ORCFile文件格式的数据更新操作，还需要提前开启事务支持。

2)Hive的运行依赖HDFS进行数据存储，默认依赖MapReduce进行数据计算。

HBase

1)HBase本身不支持SQL查询功能，需要通过与Phoenix集成实现。

2)HBase的运行依赖Zookeeper提供协调服务(配置服务、维护元数据、命名空间服务)，依赖HDFS数据存储。

4.应用场景的区别

Hive

1）Hive主要用于构建基于Hadoop平台的数据仓库，离线处理海量数据。

2）Hive提供完整的SQL实现，一般用于历史数据的分析、挖掘。

HBase

1）HBase适合用了大数据的实时查询、海量数据的存储。

2）HBase是近实时数据库，支持线上业务的实时查询。

### 第2题

手写拉链表SQL，并分析有多少个job？

参考答案：

1.拉链表定义

维护历史状态，以及最新状态数据的一种表，拉链表根据拉链粒度的不同，实际上相当于快照，只不过做了优化，去除了一部分不变的记录而已,通过拉链表可以很方便的还原出拉链时点的客户记录。

2.用户表

用户表中的数据每日既有可能新增， 也有可能修改， 但修改频率并不高， 属于缓慢变化

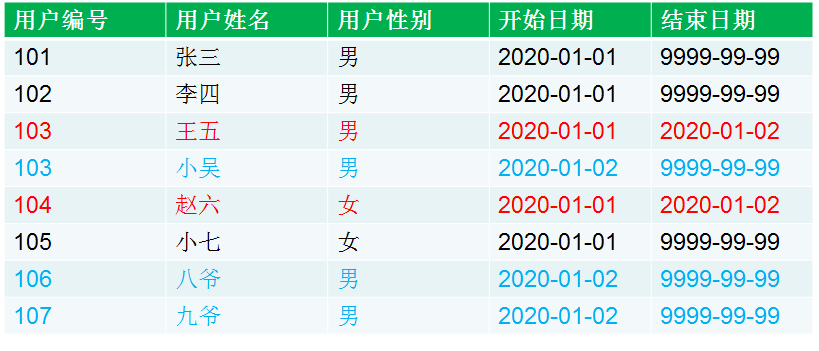
维度， 此处采用拉链表存储用户维度数据。

3.拉链表实现原理

第一步：比如在2020-01-01这一天，我们初始化用户数据到数据仓库user表，user表包含start\_time和end\_time字段用来标识该条数据的生命周期，明细数据如下所示。



第二步：在2020-01-02这一天，104用户被删除，同时增加了106和107用户，103用户的名称改成了小吴，明细数据如下所示。



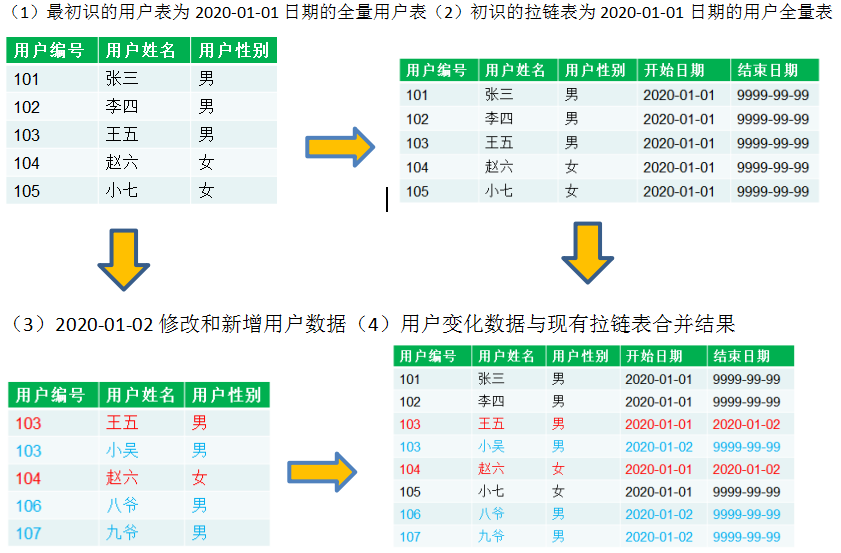
第三步：在2020-01-03这一天，107用户被删除，同时106用户名称改为八王爷，103用户名称改为吴王，并增加了108用户，明细数据如下所示。



4.拉链表使用场景

数据量有点大，表中某些字段有变化，但是变化的频率不是很高（数据缓慢变化），业务需求又需要统计这种变化状态。但每天全量一份数据，有点不太现实，不仅浪费了存储空间，有时可能业务统计也有点麻烦。这时，拉链表的作用就体现出来了，既节省空间，又满足了需求。比如用户表，用户量1亿，一年365天，每天存储一份全量用户数据，效率非常低，此时就可以使用用户拉链表。

5.拉链表实现步骤



1)原始业主用户表

drop table if exists o\_sca\_owner\_d;

create external table o\_sca\_owner\_d (

`id` string COMMENT '业主id',

`roomId` string COMMENT '业主房间号',

`ownerName` string COMMENT '业主姓名',

`ownerSex` string COMMENT '业主性别',

`ownerAge` string COMMENT '业主年龄',

`ownerIdcard` string COMMENT '业主身份证号',

`ownerTel` string COMMENT '业主联系电话',

`ownerHomeplace` string COMMENT '业主籍贯',

`ownerWorkplace` string COMMENT '业主工作地',

`create\_time` string COMMENT '创建时间',

`operator\_time` string COMMENT '操作时间'

) COMMENT '业主表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS textfile

location '/user/hive/warehouse/sca.db/o\_sca\_owner\_d/';

2)初始化拉链表

drop table if exists dim\_owner\_l;

create external table dim\_owner\_l (

`id` string COMMENT '业主id',

`roomId` string COMMENT '业主房间号',

`ownerName` string COMMENT '业主姓名',

`ownerSex` string COMMENT '业主性别',

`ownerAge` string COMMENT '业主年龄',

`ownerIdcard` string COMMENT '业主身份证号',

`ownerTel` string COMMENT '业主联系电话',

`ownerHomeplace` string COMMENT '业主籍贯',

`ownerWorkplace` string COMMENT '业主工作地',

`create\_time` string COMMENT '创建时间',

`operator\_time` string COMMENT '操作时间',

`start\_time` string COMMENT '有效开始日期',

`end\_time` string COMMENT '有效结束日期'

) COMMENT '业主拉链表表'

stored as parquet

location '/user/hive/warehouse/sca.db/dim\_owner\_l/';

3)制作临时拉链表

drop table if exists dim\_owner\_l\_tmp;

create external table dim\_owner\_l\_tmp (

`id` string COMMENT '业主id',

`roomId` string COMMENT '业主房间号',

`ownerName` string COMMENT '业主姓名',

`ownerSex` string COMMENT '业主性别',

`ownerAge` string COMMENT '业主年龄',

`ownerIdcard` string COMMENT '业主身份证号',

`ownerTel` string COMMENT '业主联系电话',

`ownerHomeplace` string COMMENT '业主籍贯',

`ownerWorkplace` string COMMENT '业主工作地',

`create\_time` string COMMENT '创建时间',

`operator\_time` string COMMENT '操作时间',

`start\_time` string COMMENT '有效开始日期',

`end\_time` string COMMENT '有效结束日期'

) COMMENT '业主拉链表表'

stored as parquet

location '/user/hive/warehouse/sca.db/dim\_owner\_l\_tmp/';

4)拉链表具体sql实现

insert overwrite table dim\_owner\_l\_tmp

select \* from

(select

id,

roomId,

ownerName,

ownerSex,

ownerAge,

ownerIdcard,

ownerTel,

ownerHomeplace,

ownerWorkplace,

create\_time,

operator\_time,

'2021-05-21' start\_time,

'9999-99-99' end\_time

from o\_sca\_owner\_d where dt='2021-05-21'

union all

select

ol.id,

ol.roomId,

ol.ownerName,

ol.ownerSex,

ol.ownerAge,

ol.ownerIdcard,

ol.ownerTel,

ol.ownerHomeplace,

ol.ownerWorkplace,

ol.create\_time,

ol.operator\_time,

ol.start\_time,

if(oso.id is not null and ol.end\_time='9999-99-99', date\_add(oso.dt,-1),

ol.end\_time) end\_time

from dim\_owner\_l ol left join

(

select

\*

from o\_sca\_owner\_d

where dt='2021-05-21'

) oso on ol.id=oso.id

)olt

order by olt.id, olt.start\_time;

5)Hive中stage与job的关联

（1）通过explain可以清晰的看到stage划分。

（2）经过自动优化之后，有些stage并不会执行，这些stage就是空的stage。

（3）基于stage划分，如何确定job数量，主要是看哪些stage会执行。只有需要执行的stage才会提交到yarn，然后生成具体的job application。

6）生成最终拉链表

临时拉链表导入最终拉链表

insert overwrite table dim\_owner\_l select \* from dim\_owner\_l\_tmp;

7）查询最终拉链表数据

select id, ownerName,start\_time, end\_time from dim\_owner\_l limit 10;

### 第3题

手写连续登入7日的用户SQL？

参考答案：

1.数据结构

1）数据格式

第一列(uid)：用户id

第二列(dt)：登录时间

第三列(status)：用户状态

2）数据示例

1,2021-05-11,1

1,2021-05-12,1

1,2021-05-13,1

1,2021-05-14,1

1,2021-05-15,1

1,2021-05-16,1

1,2021-05-17,1

1,2021-05-18,1

2.创建表

create table ulogin(

uid int, dt date, status int )

row format delimited fields terminated by ','

stored as textfile

location '/user/hive/warehouse/sca.db/ulogin/';

3.加载数据

load data local inpath '/home/hadoop/shell/data/userlogin.txt' into table ulogin;

4.具体实现

1）使用row\_number()函数在组内给每行数据编号

使用的row\_number 给日期排序

select uid, dt, row\_number() over (partition by uid order by dt) gid from ulogin where status=1;

2）日期-rownum = gid，得到结果可以作为连续登录的依据

通过date\_sub函数来获取连续登录的日期

select uid, dt, date\_sub(dt, row\_number() over (partition by uid order by dt)) gid from ulogin where status=1;

3）根据求得的gid作为分组条件，统计最终结果

通过group by count having分组统计，即可获取连续登入7日的用户

select uid,count(gid) from (select uid, dt, date\_sub(dt, row\_number() over (partition by uid order by dt)) gid from ulogin where status=1) as u group by uid,gid having count(gid)>=7;

### 第4题

Hive架构原理？

参考答案：

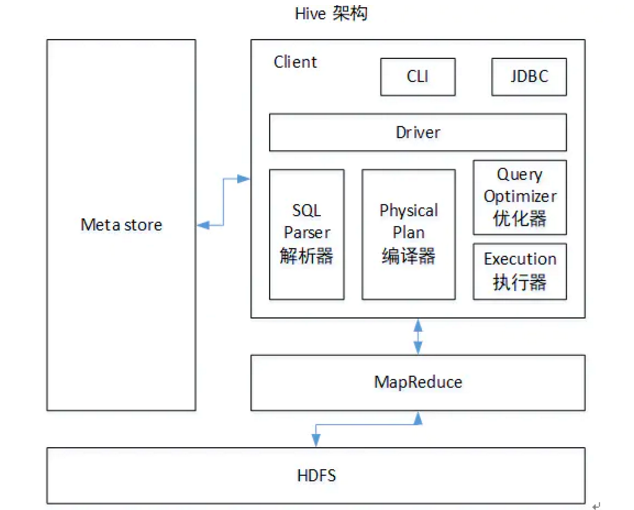
1.Hive是什么

1）Hive是由Facebook开源用于解决海量结构化日志的数据统计

2）Hive是基于Hadoop的一个数据仓库工具，可以将结构化的数据文件映射 成一张表

3）Hive供类了SQL查询功能，底层计算引擎默认为MapReduce，存储为HDFS

2.Hive架构



3.Hive优点

1）操作接口采用类SQL语法，提供快速开发的能力；

2）避免了编写MapReduce，减少开发人员的学习成本；

4）易扩展(HDFS+MapReduce：可以扩展集群规模；支持自定义函数)；

4.Hive使用场景

1）数据的离线处理；比如,日志分析,海量结构化数据离线分析;

2）Hive的执行延迟比较高，因此hive常用于数据分析的对实时性要求不高的场合；

3）Hive优势在于处理大数据，对于处理小数据没有优势，因为Hive的执行延迟比较高。

5.Hive的执行流程

1）Hive通过用户即可（如CLI/JDBC）向Hive Driver发送执行命令，目的是获取查询结果；

2）Driver拿到执行命令后，会向Complier（编译器）获取查询计划；

3）Complier生成查询计划，首先要向MetaStore获取相关的元数据信息；

4）Driver获取返回的查询计划并进行优化，然后将执行计划提交给Hadoop运行；

5）Hadoop将执行的结果返回给Driver；

6）Driver将执行结果返回给客户端。

6.Hive SQL转换成MapReduce的过程

1）Parser：将 HQL 语句转换成抽象语法树（AST：Abstract Syntax Tree）；

2）Semantic Analyzer：将抽象语法树转换成查询块；

3）Logic Plan Generator：将查询块转换成逻辑查询计划；

4）Logic Optimizer：重写逻辑查询计划，优化逻辑执行计划；

5）Physical Plan Gernerator：将逻辑计划转化成物理计划（MapReduce Jobs）；

6）Physical Optimizer：选择最佳的 Join 策略，优化物理执行计划。

### 第5题

Hive内外表的区分方法与差异？

参考答案：

1.Hive内外表区分方法

1）查看hive元数据：进入mysql中hive元数据库，查看TBLS表，查看对应的表名和表类型；

2）在hive-cli界面：desc extended tablename，查看TableType，一般在展示结果的最后显示；

3）在hive-cli界面：desc formatted tablename，一个格式化的信息，查看TableType；

2.Hive内外表差异

（1）内部表

1）hive创建表的时候默认是MANAGED\_TABLE(内部表)

2）hive创建表的时候如果指定location，当进行load操作的时候，hdfs上的原始数据会迁移到location指定的目录下；

3）删除hive内部表的时候,会将hdfs目录的数据和元数据信息全部删除；

create table if not exists temperature

(id string comment '气象站id',year string comment '年',temperature int comment '气温')

comment '天气表'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE ;

load data inpath '/weather/temperature' overwrite into table temperature ;

（2）外部表

1）删除hive表的时候只会将元数据信息删除，并不会删除HDFS上的数据；

2）生产环境中，建议使用hive外部表；

create external table if not exists external\_temperature

(id string comment '气象站id',year string comment'年',temperature int comment '气温')

comment '天气表'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE LOCATION '/user/hive/warehouse/myoutdb/external\_temperature';

load data inpath '/weather/temperature' overwrite into table external\_temperature;

### 第6题

Hive的数据模型及应用场景？

参考答案：

1.Hive的数据模型

Hive的数据模型主要有：database、table、partition、bucket四部分；

2.数据模型之database

database：相当于关系型数据库中的命名空间，作用是将数据库应用隔离到不同的数据库模式中，hive提供了create database dbname语句来建库；

3.数据模型之table

table：表是由存储的数据以及描述表的一些元数据组成。数据存储在HDFS分布式文件系统中，元数据存储在MySQL关系型数据库中；

hive常见表分类：

MANGED\_TABLE :内部表

EXTERNAL\_TABLE：外部表

4.数据模型之partition

hive的分区是根据某列的值进行粗略的划分，每个分区对应HDFS上的一个目录；

5.数据模型之bucket

要使用hive的分桶功能，首先需要打开hive对桶的开关 set hive.enforce.bucketing=true;

bucket是按照指定值进行hash，每个桶就是表目录里的一个文件。

### 第7题

Kylin与ClickHouse全面比较？

参考答案：



### 第8题

数据倾斜产生原因及处理方式？

参考答案：

1.数据倾斜定义

任务进度长时间维持在99%，查看任务监控页面，发现只有少量1个或几个reduce子任务未完成。因为其处理的数据量和其他reduce差异过大。单一reduce的记录数与平均记录数差异过大，通常可能达到3倍甚至更多，最长时长远大于平均时长。

2.数据倾斜原因

map输出数据按key Hash的分配到reduce中，由于key分布不均匀、业务数据本身的特性、建表时考虑不周、某些SQL语句本身就有数据倾斜等原因，造成的reduce上的数据量差异过大，所以如何将数据均匀的分配到各个reduce中，就是解决数据倾斜的根本所在。

3.key为空引起数据倾斜

（1）倾斜原因

join的key值发生倾斜，key值包含很多空值或是异常值。

（2）解决方案

对值为空的key进行打散，为空key赋一个随机的值，使得key值为空的数据随机均匀地分布到不同的 reducer上。

（3）测试案例

设置多个reduce格式

set mapreduce.job.reduces = 5;

两张大表join ，做全连接

select t.id,t.year,t.temperature,s.state from temperature t full join station s on nvl(t.id,rand())=s.id limit 10;

备注：nvl()为空值转换函数，rand()为随机函数

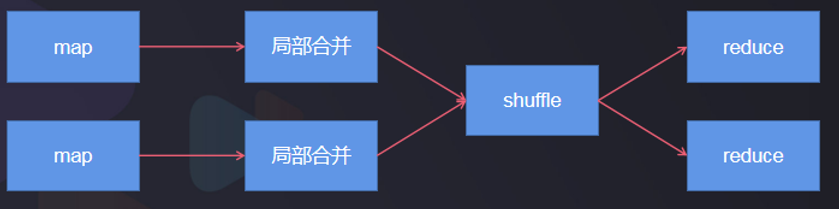
4.group by引起数据倾斜

（1）倾斜原因

默认情况下，Map 阶段同一 Key 数据分发给一个 reduce，当一个 key 数据过大时就倾斜了。

（2）解决方案

然而并不是所有的聚合操作都需要在 Reduce 端完成，很多聚合操作都可以先在 Map 端进行部分聚合，最后在 Reduce 端得出最终结果。



（3）实现方式1

1）是否在 Map 端进行聚合，默认为 True（使用Combiner局部合并）

set hive.map.aggr = true;

2）设置map端预聚合的行数阈值

set hive.groupby.mapaggr.checkinterval = 100000;

（4）实现方式2

1）有数据倾斜的时候进行负载均衡（默认是 false）

set hive.groupby.skewindata = true;

当遇到数据倾斜时，group by会启动两个MR job。第一个job会将map端数据随机输入reducer，每个reducer做部分聚合，相同的key就会分布在不同的reducer中。第二个job再将前面预处理过的数据按key聚合并输出结果，这样就起到了均衡的效果。

2）测试案例

set hive.map.aggr = true;

set hive.groupby.mapaggr.checkinterval = 100000;

set hive.groupby.skewindata = true;

select id,count(\*) from temperature group by id;

5.count distinct引起数据倾斜

（1）产生原因

count distinct聚合时存在大量特殊值，比如存在大量值为NULL或空的记录。

（2）解决方式

做count distinct时，将值为空的情况单独处理。

1）如果只是统计去重后的记录数，可以不用处理空值，先把空值过滤掉，然后在最后结果中加1即可

2）如果还包含其他计算，需要进行group by操作，先将值为空的记录单独处理，然后再跟其他计算结果union 操作。

6.join操作引起数据倾斜

（1）大表join小表

1）产生原因

业务数据本身就存在key分布不均匀的情况，一般情况会产生数据倾斜

2）解决方式

使用map join让小的维度表先进内存，在map端完成join

3）实现原理

使用map join，直接在map端就完成表的join操作，进入map端的数据都是经过split得到的，没有根据key分区这一操作，所以数据都是相对均匀地分布在每个map task中的，所以就不会产生数据倾斜。

（2）大表join大表

1）产生原因

业务数据本身的特性，导致两个表都是大表。

2）解决方式

业务消减

3）实现原理

业务数据有数据倾斜的风险，但是这些导致数据倾斜风险的key一般都是无效的，如uid为空，因为uid为空的记录是没有意义的。

所以当业务数据很大，但是数据中的大部分（一般都是80%）可能都是无效数据，那么就可以在join时过滤掉空值uid，没有了这些无效数据，自然就不存在这么大量集中的key，数据倾斜的风险就会消失。

### 第9题

统计注册用户留存天数与留存人数？

参考答案：

1.日志表

（1）用户日志临时表

CREATE TABLE `user\_log\_tmp`(

`uid` int,

`action` string,

`day` date)

row format delimited fields terminated by ','

STORED AS textfile

LOCATION '/user/hive/warehouse/mydb/user\_log\_tmp';

uid 用户id，action 用户的具体行为，day 产生日志的时间 (按天分区）;

（2）用户日志表

CREATE TABLE `user\_log`(

`uid` int,

`action` string)

PARTITIONED BY (`day` date)

row format delimited fields terminated by ','

STORED AS textfile

LOCATION '/user/hive/warehouse/mydb/user\_log';

uid 用户id，action 用户的具体行为，day 产生日志的时间 (按天分区）;

（3）加载数据

load data local inpath '/home/hadoop/shell/data/userlog.txt' overwrite into table user\_log\_tmp;

INSERT OVERWRITE TABLE user\_log PARTITION(day) SELECT uid,action,day FROM user\_log\_tmp ;

2.全量用户表

（1）建表

CREATE TABLE `all\_user`(

`uid` int,

`createtime` date)

row format delimited fields terminated by ','

STORED AS textfile

LOCATION '/user/hive/warehouse/mydb/all\_user';

uid 用户id，createTime用户第一次有记录的时间;

3.每天去重用户表

（1）建表

CREATE TABLE `daily\_user`(

`uid` int)

PARTITIONED BY (`day` date)

row format delimited fields terminated by ','

STORED AS textfile

LOCATION '/user/hive/warehouse/mydb/daily\_user';

uid 用户id，day 日期（按天分区）;

4.具体实现步骤

（1）统计获得每天的去重用户

INSERT OVERWRITE table daily\_user partition(day = '2021-05-09') SELECT DISTINCT uid FROM user\_log WHERE day = '2021-05-09';

（2）更新全量用户

INSERT INTO table all\_user (uid, createtime) SELECT u.uid, '2021-05-09' AS createtime FROM (SELECT DISTINCT uid FROM user\_log WHERE day = '2021-05-09') AS u WHERE u.uid NOT IN (SELECT uid FROM all\_user);

（3）计算用户注册时间与活动时间差值

SELECT au.uid, au.createtime, du.day, datediff(du.day, au.createtime )

FROM all\_user au INNER JOIN daily\_user du

ON au.uid = du.uid WHERE au.createtime = '2021-05-09';

（4）统计留存天数与人数

SELECT datediff(du.day, au.createtime) AS day , COUNT(du.uid) AS total

FROM all\_user au INNER JOIN daily\_user du

ON au.uid = du.uid

WHERE au.createtime = '2021-05-09'

GROUP BY datediff(du.day, au.createtime);

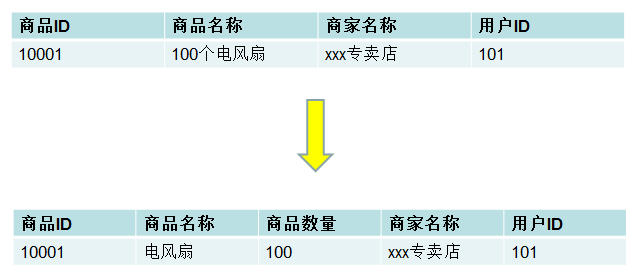
### 第10题

数据库的三范式？

参考答案：

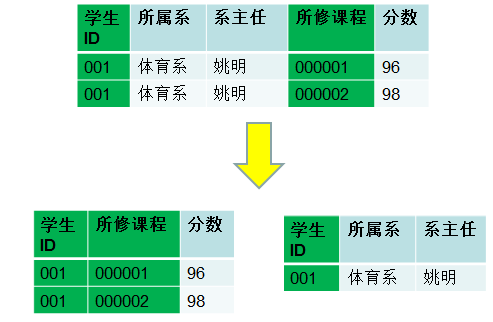
第一范式：

域都应该是原子性的， 即数据库表的每一列都是不可分割的原子数据项。



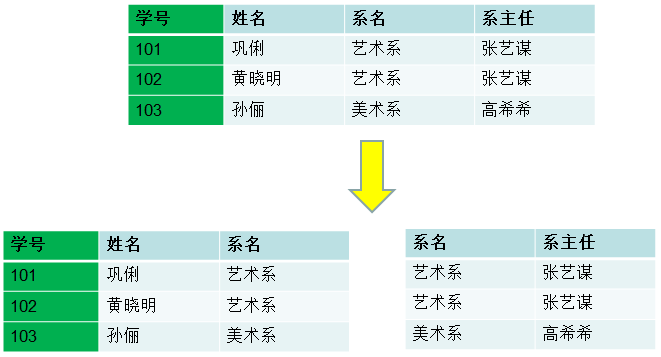
第二范式：

在1NF的基础上， 实体的属性完全依赖于主关键字，不能存在仅依赖主关键字一部分的属性。



第三范式：

在2NF的基础上， 任何非主属性不依赖于其它非主属性。只要非主键内部存在传递依赖，就不满足第三范式。



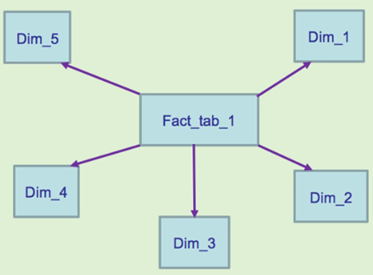
### 第11题

维度建模常用模型？

参考答案：

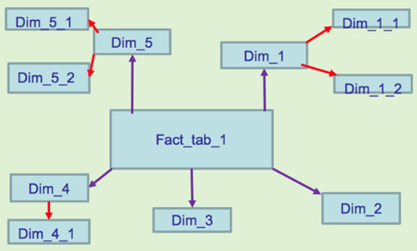
星型模型

当所有维表都直接连接到 "事实表"上时，整个图解就像星星一样，故将该模型称为星型模型.



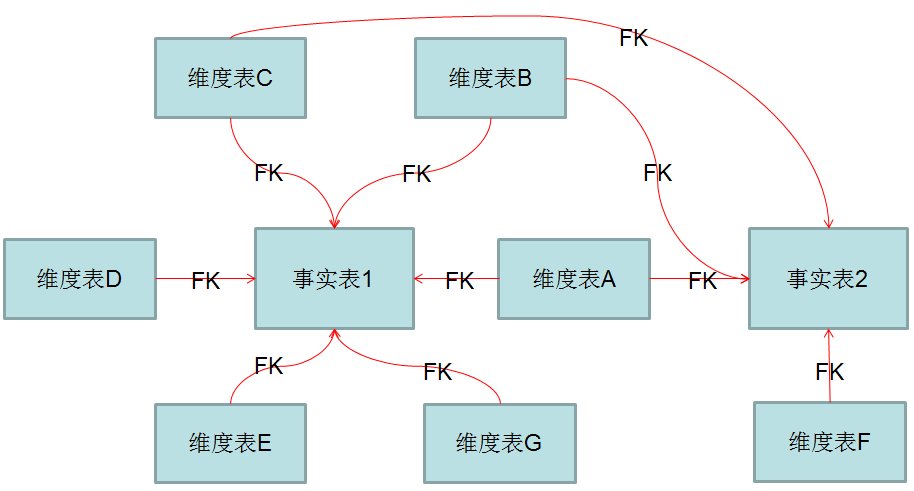
雪花模型

当有一个或多个维表没有直接连接到事实表上，而是通过其他维表连接到事实表上时，其图解就像多个雪花连接在一起，故称雪花模型。



星座模型

星座模式是由星型模式延伸而来，星型模式是基于一张事实表，而星座模式是基于多张事实表，而且共享维度表信息。



### 第12题

维度建模一般分为哪几步？

参考答案：

1.选择业务过程

通过对业务需求以及可用数据源的综合考虑，决定对哪种业务过程开展建模工作。

在业务系统中，先挑选影响大、可行性高、最感兴趣的一个或几个业务，如门禁业务和车辆进出业务，一条业务线对应一个事实表。

例如，可以选择这些业务，如门禁、车辆进出等。

2.声明粒度

数据粒度：指数据仓库中保存数据的细化程度。

声明粒度：意味着精确定义事实表中的一行数据表示什么，一般尽可能选择最小粒度，从而应对不同粒度的需求。

例如：

如在门禁表中，业主的每次开门作为门禁事实表中的一行，粒度为每次开门；每周的开门次数作为一行，粒度就是每周开门。每月的开门次数作为一行，粒度就是每月开门。

3.确定维度

事实表粒度选择完毕后，维度的选择就比较直接了，在主维度框架内，可以考虑其他维度是否可以被属性化为开门度量。如业主、日期、小区等。

维度的作用是描述业务事实，如“谁”，“何时”，“何地”，“做了什么事”。

4.确定事实

确认应该将那些事实放到事实表中，在考虑可能存在的事实时，可能会发现仍然需要调整早期的粒度声明或者维度选择。事实指的就是业务中的度量值，如订单金额，下单次数等。

事实表=维度表+度量值



### 第13题

数据集市VS数据仓库？

参考答案：

数据集市：可以理解为一种微型的数据仓库。它通常包含更少的数据，更少的主题以及更少的历史数据，属于部门级的，一般只能为某个局部范围内的管理人员服务。

数据仓库：是企业级的，能为企业各个部门的运行提供决策支持手段。

### 第14题

Oracle业务数据迁移数仓？

参考答案：

批量抽取数据库表，一般两种方式：

关系型数据库，可以通过sqoop、datax同步到hdfs中，目前几乎支持了所有的主流数据库。

用spark、MR等方式开发数据同步程序，目前主要采用该方案，实际采集的时候可以根据情况考虑全量或者增量；

### 第15题

事实表的概念？

参考答案：

每个数据仓库都包含一个或者多个事实数据表。

事实数据表可以包含业务销售数据，如现金登记事务所产生的数据，事实数据表通常包含大量的行。

一般事实表中只存放数字或者一些Flag用来统计(Count)，如收益、数量、支出等。

### 第16题

Hive分区表VS分桶表？

参考答案：

1.建表方式不同

创建分区表时，分区字段不能直接写在表结构中，通过指定分区字段和类型的方式成为表结构；

创建分桶表时，分桶表的字段必须写在表结构中，再指定分桶字段。

2.数据存储方式不同：

分区表是根据不同分区以文件夹的形式存储在HDFS上；

分桶表是以多个数据文件的形式存储在表数据目录下。

3.使用场景不同

分区表适合用于多条数据有相同字段值的情况，比如按天分区。

分桶表适合用于数据字段没有重复值的情况，或者分区太多更不利于查询的情况，比如用户表uid字段。

### 第17题

数仓架构分层及每层作用？

参考答案：

1.数仓为什么要分层

（1）把复杂问题简单化，将复杂的任务分解成多层来完成，每一层只处理简单的任务，方便定位问题。

（2）减少重复开发，规范数据分层，通过中间层数据，能够减少很多重复计算工作，提高计算结果的复用性。

1. 隔离原始数据，使真实数据与统计数据隔离开，ODS之后层级的数据处理，无论出现数据的异常还是数据的敏感性，都可以重新从ODS再计算。

2.数仓架构分层

2.1ODS层

ODS(Operational Data Store) 操作性数据，即原始数据层，是数据仓库第一层数据，是最接近数据源中数据的一层，作为数据库到数据仓库的一种过渡。数据源中的数据，经过ETL之后装入ODS层，ODS的数据结构一般与数据来源保持一致，便于减少ETL的工作复杂性。

2.2DWD层

DWD(data warehouse detail) 是明细数据层，该层的表结构和粒度与原始表保持一致，不过需要对ODS层数据进行清洗、维度退化、脱敏等，最终得到的数据是干净的，完整的，一致的数据。

2.3DWS层

DWS（data warehouse service）：服务数据层，基于DWD层上的明细数据，按天轻度汇总成某一个主题域的服务数据，一般是宽表。

DWS层统计各个主题对象的当天行为，以及一些业务明细数据，服务于DM层的某一个主题域宽表。例如每日业主行为，每日车辆行为。

2.4ADS层

ADS（Application Data Store）层：数据应用层，也即是出报表结果。以DW为基础，为各种统计报表提供汇总数据，然后将分析汇总数据同步到RDS数据库。

RDS是OLTP数据库。RDS用于支撑日常的业务活动，市场上典型产品是Mysql、Oracle。

2.5DM层

数据集市层DM(data market)，基于DW上的基础数据，整合汇总成分析某一个主题域的服务数据，一般是宽表，用于提供后续的业务查询，OLAP分析，数据分发等。

数据集市，以某个业务应用为出发点而建设的局部DW，DM只关心自己需要的数据，不会全盘考虑企业整体的数据架构和应用，每个应用有自己的DM。

2.6DM VS ADS

狭义上，数据集市，可以理解为数据仓库中为用户提供数据支撑的应用层，比如ADS层。

广义上，数据集市，所有以主题划分的数据仓库中可供查阅的都可以成为数据集市，包括DWD,DWS,ADS层，甚至包括从hadoop中同步到RDS的数据都可以成为数据集市。

### 第18题

数仓仓库 VS 数据库？

参考答案：

1.数据库定义

数据库是按照数据结构来组织、存储和管理数据的建立在计算机存储设备上的仓库。

数据库是长期存储在计算机内、有组织的、可共享的数据集合。

数据库中的数据指的是以一定的数据模型组织、描述和存储在一起、具有尽可能小的冗余度、较高的数据独立性和易扩展性的特点并可在一定范围内为多个用户共享。

2.数据仓库定义

数据仓库是面向主题的，集成的，相对稳定的反映历史变化的数据集合，用于支持管理决策。

面向主题的：在较高层次上将企业信息系统的数据综合归并进行分析利用的抽象的概念。每个主题基本上对应一个相应的分析领域。

集成的：企业级数据，同时数据要保持一致性、完整性、有效性、精确性。

稳定的：从某个时间段来看是保持不变的，没有更新操作、删除操作，以查询分析为主。

变化的：反应历史变化。

3.数据仓库VS数据库

面向业务的数据库常称作OLTP，面向分析的数据仓库亦成为OLAP。



### 第19题

如何构建数据仓库？

参考答案：

1.系统分析，确定主题

通过与业务部门的交流，了解建立数仓要解决的问题，确认各个主题下的查询分析要求

2.选择满足数据仓库系统要求的软件平台。

选择合适的软件平台，包括数据库、建模工具、分析工具等

3.建立数据仓库的逻辑模型。

确定建立数据仓库逻辑模型的基本方法，基于主题视图，把主题视图中的数据定义转到逻辑数据模型中

1. 逻辑数据模型转换为数据仓库数据模型

5.数据仓库数据模型优化

随着需求和数据量的变化进行调整

6.数据清洗转换和传输

业务系统中的数据加载到数据仓库之前，必须进行数据的清洗和转换，保证数据仓库中数据的一致性。

7.开发数据仓库的分析应用

满足业务部门对数据进行分析的需求。

8.数据仓库的管理

包括数据库管理和元数据管理。

### 第20题

数据中台VS数据仓库VS大数据平台？

参考答案：

1.什么是数据中台

数据中台是指通过数据技术，对海量数据进行采集、计算、存储、加工，同时统一标准和口径。数据中台把数据统一之后，会形成标准数据，再进行存储，形成大数据资产层，进而为客户提供高效服务。

这些服务和企业的业务有较强的关联性，是企业所独有且能复用的，它是企业业务和数据的积淀，其不仅能降低重复建设，减少烟囱式协作的成本，也是差异化竞争的优势所在。

数据中台通过整合公司开发工具、打通全域数据、让数据持续为业务赋能，实现数据平台化、数据服务化和数据价值化。数据中台更加侧重于“复用”与“业务”。

2.数据中台VS数据仓库VS大数据平台

2.1基础能力上的区别

数据平台：提供的是计算和存储能力。

数据仓库：利用数据平台提供的计算和存储能力，在一套方法论指导下建设的一整套的数据表。

数据中台：包含了数据平台和数据仓库的所有内容，将其打包，并且以更加整合以及更加产品化的方式对外提供服务和价值。

2.2业务能力上的区别

数据平台：为业务提供数据主要方式是提供数据集。

数据仓库：相对具体的功能概念是存储和管理一个或多个主题数据的集合，为业务提供服务的方式主要是分析报表

数据中台：企业级的逻辑概念，提供企业数据产生价值的能力，为业务提供服务的主要方式是数据API

### 第21题

数仓最重要的作用?

参考答案：

数据集成

（1）个人认为是数据集成。

（2）企业的数据通常是存储在多个异构数据库中，要进行分析必须先要对数据进行一致性整合。

（3）集成整合后才可以对数据进行分析，挖掘数据潜在的价值。

### 第22题

缓慢变化维度的常用处理方式？

参考答案：

1.维度分类

1.1无变化维度

比如一个人的身份证号、姓名、性别等信息属于不变的信息

1.2缓慢变化维度（slowly changing dimensions）

比如一个人的政治面貌、婚姻状态等信息属于缓慢变化的信息

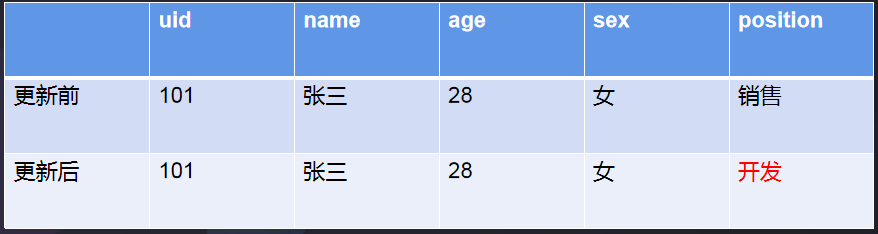
1.3剧烈变化维度

比如一个人的工作经历、工作单位等信息属于剧烈变化的信息

2.SCD常用的处理方式

2.1完全不记录历史变化信息

在ETL将数据载入SCD的时候，对于会产生变化的属性值直接覆盖。



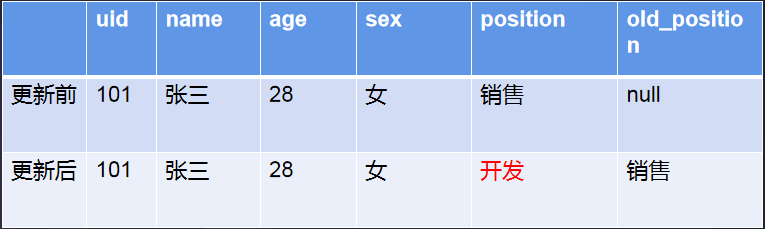
2.2通过添加记录将每一次变化都记录到SCD中

在ETL将数据载入SCD的时候，对于会产生变化的属性值保留，同时新增一条记录。



2.3添加相应的历史字段，记录最近的变化而非全部变化

如果字段发生了变化，通过添加历史字段记录最近的变化，更久之前的变化就无法追溯了。



### 第23题

元数据的理解？

参考答案：

狭义来讲：用来描述数据的数据。

广义来讲：除了业务逻辑直接读写处理的业务数据，所有其他用来维护整个系统运转所需要的数据，都可以较为元数据。

定义：元数据metadata是关于数据的数据。在数仓系统中，元数据可以帮助数据仓库管理员和数据仓库开发人员方便的找到他们所关心的数据；元数据是描述数据仓库内部数据的结构和建立方法的数据。

按照用途可分为：技术元数据、业务元数据。

1.技术元数据

存储关于数据仓库技术细节的数据，用于开发和管理数据仓库使用的数据。

数据仓库结构的描述，包括数据模式、视图、维、层次结构和导出数据的定义，以及数据集市的位置和内容。

业务系统、数据仓库和数据集市的体系结构和模式。

由操作环境到数据仓库环境的映射，包括元数据和他们的内容、数据提取、转换规则和数据刷新规则、权限等。

2.业务元数据

从业务角度描述了数据仓库中的数据，他提供了介于使用者和实际系统之间的语义层，使不懂计算机技术的业务人员也能读懂数仓中的数据。

企业概念模型：表示企业数据模型的高层信息。整个企业业务概念和相互关系。以这个企业模型为基础，不懂sql的人也能做到心中有数。

多维数据模型。告诉业务分析人员在数据集市中有哪些维、维的类别、数据立方体以及数据集市中的聚合规则。

业务概念模型和物理数据之间的依赖。业务视图和实际数仓的表、字段、维的对应关系也应该在元数据知识库中有所体现。

### 第24题

如何确定数仓主题域？

参考答案：

1.主题

主题是在较高层次上将数据进行综合、归类和分析利用的一个抽象概念，每一个主题基本对应一个宏观的分析领域。在逻辑意义上，它是对企业中某一宏观分析领域所涉及的分析对象。

面向主题的数据组织方式，就是在较高层次上对分析对象数据的一个完整并且一致的描述，能刻画各个分析对象所涉及的企业各项数据，以及数据之间的联系。

主题是根据分析的要求来确定的。

2.主题域

从数据角度看（集合论）

主题域通常是联系较为紧密的数据主题的集合。可以根据业务的关注点，将这些数据主题划分到不同的主题域。主题域的确定由最终用户和数仓设计人员共同完成。

从需要建设的数仓主题看（边界论）

主题域是对某个主题进行分析后确定的主题的边界。

数仓建设过程中，需要对主题进行分析，确定主题所涉及到的表、字段、维度等界限。

### 第25题

如何控制数据质量？

参考答案：

数据质量控制

校验机制：每天进行数据量的比对 select count(\*) from xxx，早发现，早修复。

数据内容的比对：对数据抽样比对。

复盘：每月做一次全量。

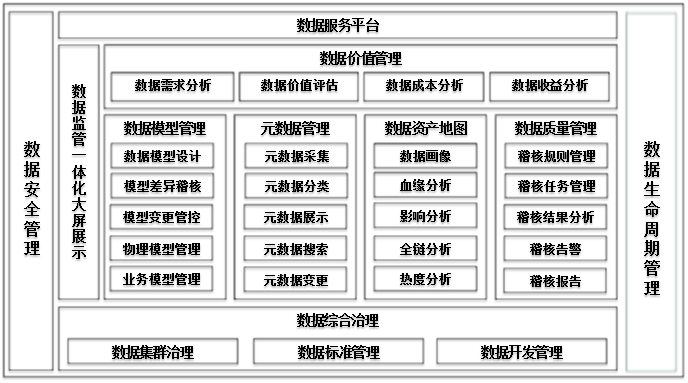
### 第26题

如何做数据治理？

参考答案：

数据治理不仅需要完善的保障机制，还需要理解具体的治理内容，比如数据应该怎么进行规范，元数据该怎么来管理，每个过程需要那些系统或者工具来配合？

数据治理领域包括但不限于以下内容：数据标准、元数据、数据模型、数据分布、数据存储、数据交换、数据生命周期管理、数据质量、数据安全以及数据共享服务。



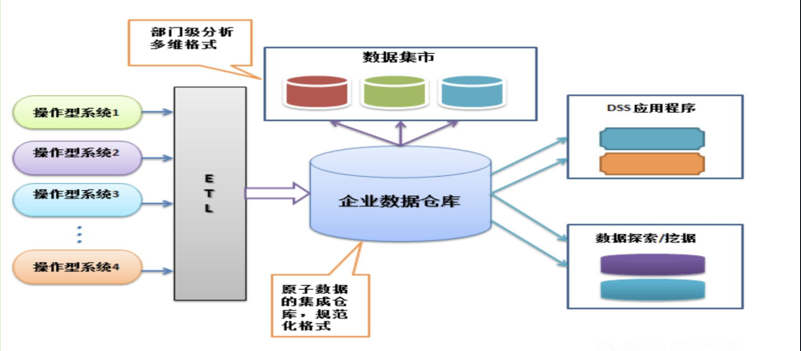
### 第27题

数仓模型设计思路？

参考答案：

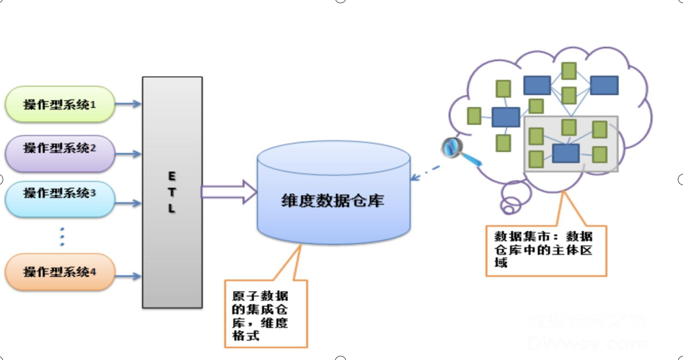
自上而下（偏数据驱动）

Bill Inmon推崇自上而下的方式，一个企业建立唯一的数据中心，数据是经过整合、清洗、去掉脏数据、标准的、能够提供统一的视图。要从整个企业的环境入手，建立数据仓库，要做很全面的设计。



自下而上（偏业务驱动）

Ralph Kimball推崇自下而上的方式，认为数据仓库应该按照实际的应用需求加载需要的数据，不需要的数据不要加载到数据仓库中。这种方式建设周期短，用户能很快看到结果。



### 第28题

为什么需要数仓建模？

参考答案：

数仓建模需要按照一定的数据模型，对整个企业的数据进行采集，整理，提供跨部门、完全一致的报表数据。

合适的数据模型，对于大数据处理来讲，可以获得得更好的性能、成本、效率和质量。良好的模型可以帮助我们快速查询数据，减少不必要的数据冗余，提高用户的使用效率。

数据仓库为企业决策，提供数据支持。基于数据仓库对数据进行分析，能够助力企业改进业务流程、控制成本、提高产品质量。

### 第29题

OLAP和OLTP模型的选择？

参考答案：

OLTP系统是操作事物型系统，主要数据操作是随机读写，主要采用满足3NF的实体关系模型存储数据，在事物处理中解决数据的冗余和一致性问题。

OLAP系统是分析型系统，主要数据操作是批量读写，不需要关注事务处理的一致性，主要关注数据的整合，以及复杂大数据量的查询和处理的性能。

### 第30题

事实表的分类？

参考答案：

1.事务事实表

事务事实表记录的是事务层面的事实，保存的是最原子的数据，也称“原子事实表”。事务事实表中的数据在事务事件发生后产生，数据的粒度通常是每个事务记录一条记录。

2.周期快照事实表

以具有规律性的、可预见的时间间隔来记录事实。它统计的是间隔周期内的度量统计，每个时间段一条记录，是在事务事实表之上建立的聚集表。比如时间周期是1周，那么这个周期快照事实表记录着一周对于某个度量值的统计值。

3.累计快照事实表

累积快照表记录的不确定的周期的数据。代表的是完全覆盖一个事务或产品的生命周期的时间跨度，通常具有多个日期字段，用来记录整个生命周期中的关键时间点。例如订单累计快照事实表会有付款日期，发货日期，收货日期等时间点。

事务事实表中一条交易记录会每天有一条数据来记录整个交易过程；而累积快照事实表只会有一条记录，数据会一直更新直到过程结束。

### 第31题

数仓架构演进

参考答案：

1.经典数仓架构：使用传统工具（如MySQL、Oracle）来建设数仓。

2.离线大数据架构：开始使用大数据工具（如Hive）来替代经典数仓中的传统工具。

3.Lambda架构：在离线大数据架构的基础上，使用流处理技术（如Spark、Flink）直接完成实时性较高的指标计算。

4.Kappa架构：实时处理变成了主要的部分，出现了以实时处理为核心的kappa架构。

### 第32题

开源OLAP解决方案？

参考答案：

Persto、SparkSQL、Impala等MPP架构和ROLAP的引擎

Druid和Kylin等预处理架构

ElasticSearch搜索引擎架构

ClickHouse列式数据库

### 第33题

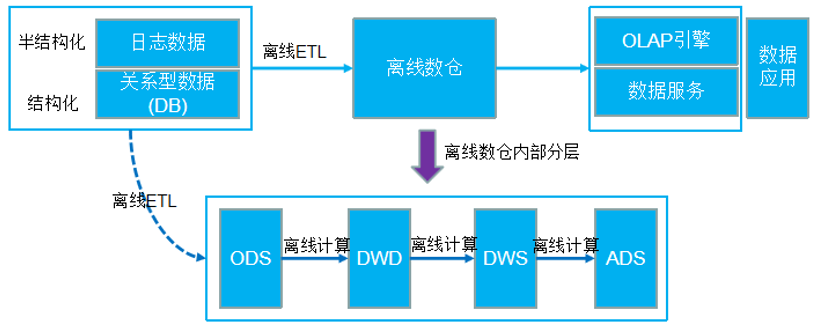
离线数仓架构？

参考答案：

离线大数据架构的特点：

（1）数据源通过离线的方式导入到离线数仓中

（2）数据处理采用MapReduce、Hive、SparkSQL等离线计算引擎



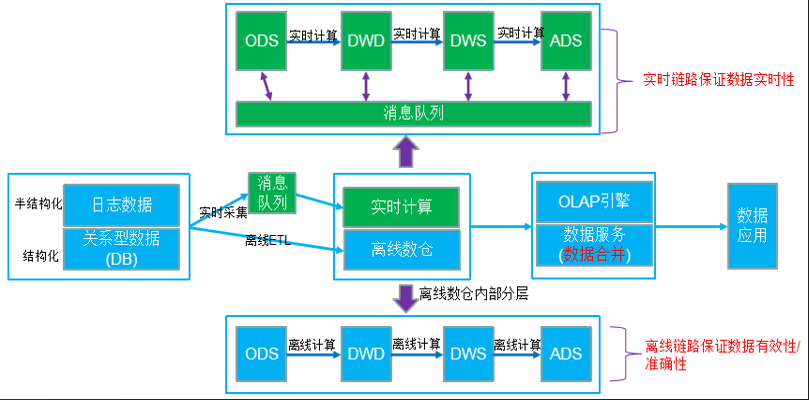
### 第34题

Lambda数仓架构？

参考答案：

在离线数仓的基础上增加了实时计算的链路，并对数据源进行流式改造，实时计算去订阅消息队列，并推送到下游的数据服务中去。

Lambda架构问题：同样的需求需要开发两套一样的代码；资源占用增多。



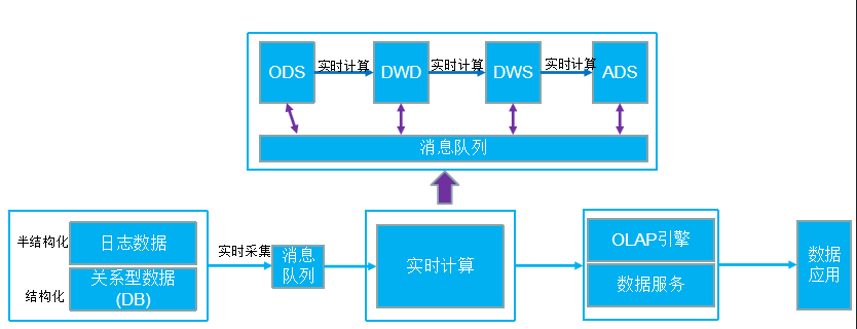
### 第35题

Kappa数仓架构？

参考答案：

kappa架构可以认为是lambda架构的简化版，移除了lambda架构中的批处理部分。在kappa架构中，需求修改或者历史数据重新处理都通过上游重放完成。

kappa架构最大的问题是流式重新处理历史数据的吞吐能力会低于批处理，但可以通过增加计算资源来弥补。



### 第36题

数仓建模方法？

参考答案：

数仓建模模型

（1）ER模型

（2）维度模型

（3）Data Vault模型

（4）Anchor模型

数仓建模常用的是维度模型和ER模型。