**面试总结**

**1、银行存款类型：**

活期、定期、定活两便存款、通知存款、大额存单、结构性存款

**2、银行贷款类型：**

委托贷款、信用贷款、担保贷款、票据贴现、综合授信、信用担保贷款、买方贷款

1. **shell使用方法**

#!/bin/bash

##hive -e 意思是到hive中去执行一个命令，命令写在""里面

#date 当前时间，输出的时候加上当前时间

echo "程序开始,时间："$(date)

hive -e "insert overwrite table mydb.dept\_avg\_sal

select

deptno,

avg(sal) as avg\_sal,

current\_timestamp as etl\_time

from mydb.emp

group by deptno"

echo "程序结束,时间："$(date)

在shell脚本中添加输出 echo ""---输出语句

---然后把这个shell脚本上传到hadoop那台机器上面

---在执行之前先授权，chmod +x dept\_avg\_sal.sh

---然后执行：

dept\_avg\_sal.sh

---重点：然后查看数据

select \* from mydb.dept\_avg\_sal;

1. **Hive抽取数据**

--字段名称 类型

int 数字

string 字符

date 日期

double 小数

create table IF NOT EXISTS mydb.emp(

empno int,

ename string,

job string,

mgr int,

hiredate string,

sal double,

comm double,

deptno int

)

COMMENT 'emp'

row format delimited

fields terminated by ',';

---导入之前先创建一个目录，

命令是：mkdir mydir

---通过xftp上传到hadoop-master的mydir文件夹下

---然后导入到hive中，也就是hadoop中

--在hive里执行：

load data local inpath '/root/mydir/emp.txt' overwrite into table mydb.emp;

Hive不支持一条一条的用insert语句进行插入操作，也不支持update的操作。

数据是以从其他表查询或load的方式，加载到建立好的表中。数据一旦导入，则不可修改。

1. **linux 指令**

linux系统 命令：

##查看目录里的内容：ls 列出

##切换目录： cd / 切换到根目录

 cd 切换到桌面

 cd /opt 进入opt目录

 cd .. 返回上一级目录

##查看机器的内存： free -h

##查看存储空间： df -h

##查看机器的ip： ifconfig；

##查看文件 vi

1. **存储过程和自定义函数的区别**

一、存储过程用户在数据库中完成特定操作或者任务（如插入，删除等），函数用于返回特定的数据。

二、存储过程声明用procedure，函数用function。

三、存储过程不需要返回类型，函数必须要返回类型。

四、存储过程可作为独立的pl-sql执行，函数不能作为独立的plsql执行，必须作为表达式的一部分。

五、存储过程只能通过out和in/out来返回值，函数除了可以使用out，in/out以外，还可以使用return返回值。

六、sql语句（DML或SELECT)中不可用调用存储过程，而函数可以。

**7、hive和sql的区别**

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为一张数据库表，并提供类SQL查询功能。Hive利用HDFS，也就是分布式文件存储系统来存储数据，利用MapReduce查询数据。

Hive 和数据库除了拥有类似的查询语言，再无类似之处。

1）数据存储位置

Hive 存储在 HDFS 。数据库将数据保存在块设备或者本地文件系统中。

2）数据更新

Hive中不建议对数据的改写。而数据库中的数据通常是需要经常进行修改的，

3）执行延迟

Hive 执行延迟较高。数据库的执行延迟较低。当然，这个是有条件的，即数据规模较小，

当数据规模大到超过数据库的处理能力的时候，Hive的并行计算显然能体现出优势。

4）数据规模

Hive支持很大规模的数据计算；数据库可以支持的数据规模较小。

**8、过往简述**

之前毕业刚开始做的是跨境电商的一个项目，主要通过分析亚马逊，ebay、速卖通电商平台的订单，通过对日浏览量、日点击量、用户搜索关键字、链接销售、排名、竞品销售排名、物流、采购、仓储、毛利润等全面的数据统计分析，为企业运营决策做数据支撑;

**9、一个引擎：spark**

--spark就是一个基于内存计算的高效的查询引擎，可以用它查询hive中的表

spark-sql的使用：

1、把hive配置文件hive-site.xml拷贝到Spark的conf目录下

命令是：

备注：cp 复制拷贝的意思

cp /opt/apache-hive-2.1.0-bin/conf/hive-site.xml /opt/spark-2.4.0-bin-hadoop2.7/conf

2、从/opt/apache-hive-2.1.0-bin/lib把mysql-connector-java-5.1.39-bin.jar拷贝到Spark的jars目录下

命令是：

cp /opt/apache-hive-2.1.0-bin/lib/mysql-connector-java-5.1.39-bin.jar /opt/spark-2.4.0-bin-hadoop2.7/jars

3、新打开一个窗口开启hive元数据服务：

hive --service metastore &

4、进入 spark-sql:

spark-sql

显示数据库：

>show databases;

5、使用spark-sql执行以上需求并对比。

测试：

select deptno,count(1) from mydb.emp group by deptno;

6、修改脚本：原来的语句是用hive -e 运行的，改动为spark-sql -e 运行。

---编写shell脚本：

#!/bin/bash

spark-sql -e ""

1. **Sqoop工具**

是一个用来将Hadoop和关系型数据库中的数据相互转移的工具，

可以将一个关系型数据库（例如 ： MySQL ,Oracle ,Postgres等）中的数据导进到Hadoop的HDFS中，

也可以将HDFS的数据导进到关系型数据库中。

(1) 创建ods

create database ods;

(2)把数据利用sqoop工具导入ods

备注：为了保证数据导入不出问题

使用

--delete-target-dir

每次先把产生的临时目录删掉

-----user表

sqoop import \

--connect jdbc:mysql://192.168.0.100:3306/order \

--username ds \

--password dstest \

--table user \

--hive-import \

--hive-table ods.ods\_user \

--hive-overwrite \

--fields-terminated-by '\t' \

--delete-target-dir

1. **索引**

---查询的执行原理：

1.提交查询语句给Oracle数据库

2.检索缓存区，缓存区会把之前的执行计划暂时保留下来。若语句和缓存区中存在的语句一模一样（多一个空格都不行），

则会减少很多解析时间，若缓存区没有，那么执行3

3.检查语法

4.检查表、字段是否存在，检查查询权限

5.设计执行计划

---创建索引

CREATE [UNIQUE] INDEX index\_name ON table\_name(column\_name[,column\_name…])

select \* from emp where ename='SCOTT'

----创建普通索引

create index idx\_ename on emp(ename);

---删掉索引

drop index idx\_ename;

----创建唯一类型的索引

create UNIQUE index idx\_ename on emp(ename);

---组合索引

select \* from emp where ename='SCOTT' and job='SALESMAN'

create index idx\_ename\_job on emp(ename,job);

---位图索引

create bitmap index idx\_job on emp(job);

---基于函数的索引

select \* from emp where lower(ename)='scott'

create index idx\_ename\_lower on emp(lower(ename));

1. **锁表**
2. 锁表的原理：数据库使用独占式封锁机制，当执行上面的语句时，对表进行锁住，直到发生commite 或者 回滚 或者退出数据库用户
2、主要的锁表原因有：
（1） A程序执行了对 tableA 的 insert ，并还未 commite时，B程序也对tableA 进行insert 则此时会发生资源正忙的异常 就是锁表
（2）锁表常发生于并发而不是并行（并行时，一个线程操作数据库时，另一个线程是不能操作数据库的，cpu 和i/o 分配原则）
3、减少锁表的概率方法：
（1）减少insert 、update 、delete 语句执行 到 commite 之间的时间。具体点批量执行改为单个执行、优化sql自身的非执行速度
（2）如果异常对事物进行回滚；
3. **主键索引和唯一索引的区别**

主键是一种约束，唯一索引是一种索引，两者在本质上是不同的。
主键创建后一定包含一个唯一性索引，唯一性索引并不一定就是主键。
唯一性索引列允许空值，而主键列不允许为空值。
主键列在创建时，已经默认为空值 + 唯一索引了。
主键可以被其他表引用为外键，而唯一索引不能。
一个表最多只能创建一个主键，但可以创建多个唯一索引。
主键更适合那些不容易更改的唯一标识，如自动递增列、身份证号等。
在 RBO 模式下，主键的执行计划优先级要高于唯一索引。 两者可以提高查询的速度。

1. **增量同步**

CREATE OR REPLACE PROCEDURE sp\_dw\_orders\_i(P\_START\_DATE VARCHAR2,

 P\_END\_DATE VARCHAR2)

IS

/\*----------------程序说明----------------

创建人：

创建时间：

修改人：

修改时间：

功能描述：增量更新dw\_orders（订单事实表）

-------------------------------------------\*/

 V\_START\_DATE DATE := TO\_DATE(P\_START\_DATE, 'YYYY-MM-DD HH24:MI:SS');

 V\_END\_DATE DATE := TO\_DATE(P\_END\_DATE, 'YYYY-MM-DD HH24:MI:SS');

 v\_sp\_name varchar2(40) := 'sp\_dw\_orders\_i';

 v\_sp\_status varchar2(10) := 'running';

 v\_s\_time date := sysdate;

 v\_f\_time date;

 v\_sp\_desc varchar2(500) := '开始跑数';

BEGIN

 sp\_tab\_log(v\_sp\_name, v\_sp\_status, v\_s\_time, v\_f\_time, v\_sp\_desc);

 ---删除这段时间的数据

 DELETE FROM dw\_orders where createtime between v\_START\_DATE and V\_END\_DATE;

 COMMIT;

 --再插入这段时间的数据

 insert into dw\_orders

 (od\_id,

 order\_id,

 user\_id,

 no,

 items\_id,

 items\_num,

 note,

 createtime,

 udt)

 SELECT ol.id,

 os.id,

 os.user\_id,

 os.no,

 ol.items\_id,

 ol.items\_num,

 os.note,

 os.createtime,

 SYSDATE

 FROM ods.ods\_orders os

 JOIN ods.ods\_orderdetail ol

 ON os.id = ol.orders\_id

 where os.createtime between v\_START\_DATE and V\_END\_DATE;

 COMMIT;

 v\_sp\_status := 'finished';

 v\_f\_time := sysdate;

 v\_sp\_desc := '跑数完成';

 sp\_tab\_log(v\_sp\_name, v\_sp\_status, v\_s\_time, v\_f\_time, v\_sp\_desc);

EXCEPTION

 WHEN OTHERS THEN

 v\_sp\_status := 'error';

 v\_f\_time := sysdate;

 v\_sp\_desc := SQLERRM;

 sp\_tab\_log(v\_sp\_name, v\_sp\_status, v\_s\_time, v\_f\_time, v\_sp\_desc);

END;