大数据架构与应用操作题

1. HDFS 操作题

假设 Hadoop 的安装目录为“/usr/local/hadoop”

前提条件

题目不是很严谨，需要确定以下条件才能正常完成，如果考到了建议先说明一下。

假设已正确配置 PATH 环境变量

题目中给了 Hadoop 的安装目录，但没有说明环境变量的配置情况。如果没有配置，则每次执行 HDFS 相关命令前，都要执行 `cd /usr/local/hadoop/bin` 命令**进入 bin 目录**，然后通过相对路径**执行当前目录下的二进制文件 hdfs**（`./hdfs dfs ...`），从而执行 HDFS 相关命令；或者，直接通过 `/usr/local/hadoop/bin/hdfs dfs` 来执行 HDFS 相关命令。不管是哪种方法，都是比较繁琐的。

一般我们都会直接将 Hadoop 的 **bin 目录添加到** `PATH` **环境变量**中，这样就可以在任意目录直接执行 `hdfs dfs` 命令，更方便一些。因此，我们可以**假设 bin 目录已经添加到** `PATH` **环境变量中。**

假设所有题目的用户均为 hadoop

HDFS 默认情况下具有严格的权限控制，只有**目录所有者**和**超级用户**才有权访问对应的用户目录。在第一题中以 hadoop 用户创建用户目录，默认情况下只有 hadoop 用户可以操作该目录，而其他用户是无法操作的。这时，还需要额外执行 `hdfs dfs -chmod 777 /user/hadoop` 命令来设置该目录的权限，允许所有用户读写。

因此，我们应该**假设使用 hadoop 用户为整道题的大前提**，而不只是第一题的小前提。在这种情况下，不需要额外设置权限，也可以**使用相对路径简化命令**（**相对路径默认为用户目录**，例如，这种情况下的相对路径 `.` 的实际路径为 `/user/hadoop`），感觉这样更符合老师出题的意图。

(1) 为 hadoop 用户在 HDFS 中创建用户目录“/user/hadoop”。

```
hdfs dfs -mkdir -p /user/hadoop
```

这里使用 `mkdir` 命令来**创建目录**，其用法与类Unix系统中的 `mkdir` 命令类似。包括 `mkdir` 在内的**所有 HDFS 命令前面都要有连字符** `-`，不要忘记哦~

值得注意的是，这里使用了 `-p` 参数，表示**递归创建目录。**如果父目录不存在，不使用 `-p` 参数会报错；如果父目录已存在了，不需要递归创建，加上这个参数也不会报错。因此，最好还是加上。

当然啦，大多数情况下在格式化节点时，会**默认创建** `/user` **目录**，因此一般来说不加 `-p` 参数也是可以的。实际上，如果以 hadoop 用户格式化节点，`/user/hadoop` 目录也是默认创建好的，我们再手动创建是会报错的。这里只是考察一下基本命令，题目不是很严谨，就不深究啦。

(2) 在 HDFS 的目录“/user/hadoop”下，创建 input 文件夹；并将 Linux 系统本地的“~/input/test.txt”文件上传到上述 HDFS 用户目录的 input 文件夹中。

```
hdfs dfs -mkdir input
hdfs dfs -put ~/input/test.txt input/
```

第一条命令使用 `mkdir` 命令来**创建目录**；第二条命令使用 `put` 命令来将**本地文件系统中的文件上传到 HDFS** 上，**第一个参数**为源文件路径（即本地文件路径），**第二个参数**为目标文件路径（即要放到 **HDFS 上的文件路径**）。这里有两点值得注意：

- 基于前面的假设，以 hadoop 用户执行该命令，可以直接使用相对路径。例如，第一条命令在 HDFS 上创建的目录绝对路径为 `/user/hadoop/input`.
- 这里目标文件路径省略了文件名 `test.txt`，因为源文件和目标文件的名称是一样的。那么 `input/` 后面的斜杠 `/` 可以省略嘛？答案是**在 input 目录存在的情况下是可以省略的。**如果 input 目录不存在，又省略了斜杠会怎样呢？可以猜一下。答案是会直接上传到用户目录，文件名为 input，而不是提示 input 目录不存在.

(3) 将 HDFS 中 input 文件夹中 test.txt 文件的内容输出到终端中。

```
hdfs dfs -cat input/test.txt
```

这里使用 `cat` 命令来**将文件的内容输出到控制台。**值得注意的是，它与类Unix系统中的 `cat` 命令类似，可以**指定多个参数来输出多个文件**（例如 `hdfs dfs -cat 1.txt 2.txt`），也可以**输出目录中的所有文件内容**（例如 `hdfs dfs -cat output/*`），这里只考察了输出单个文件的内容。

(4) 删除 HDFS 中 input 文件夹中 test.txt 文件。

```
hdfs dfs -rm input/test.txt
```

这里使用 `rm` 命令来**删除文件。**值得注意的是，它与类Unix系统中的 `rm` 命令不同，是**没有** `-f` **参数**的。

如果要**删除目录**，要**加上** `-r` **参数**，表示递归删除。例如，执行 `hdfs dfs rm -r input` 命令删除 input 目录，注意不要习惯性地把参数写成 `-rf`！

2、 Hbase 的 Shell 操作

(1) 列出 HBase 所有的表的相关信息。

```
hbase> list
```

(2) 创建了一个 student 表，属性有: name, sex, age, dept, course.

```
hbase> create 'student', 'name', 'sex', 'age', 'dept', 'course'
```

注意：HBase 的表中会有一个系统默认的属性作为行键，无需自行创建。

(3) 为 student 表添加了学号为 22001，名字为 SuHai 的一行数据，其行键为 22001.

```
hbase> put 'student', '22001', 'name', 'Su Hai'
```

注意：一次只能为一行数据的一个列添加一个数据。如果要为一行数据添加多个列的数据，需要分开执行命令。

(4) 即为 22001 行下的 course 列族的 math 列添加了一个数据 80.

```
hbase> put 'student', '22001', 'course:math', '80'
```

(5) 在终端打印出表 student 的所有记录数据。

```
hbase> scan 'student'
```

(6) 删除 student 表中 22001 行下的 sex 列的所有数据。

```
hbase> delete 'student', '22001', 'sex'
```

(7) 删除了 student 表中的 95001 行的全部数据。

```
hbase> deleteall 'student', '95001'
```

(8) 删除表 student.

```
hbase> disable 'student'
hbase> drop 'student'
```

注意：需要删除 HBase 表格时，需要先禁用表格，避免在删除过程中数据写入冲突或错误。同理，在对表结构进行变更、备份或恢复 HBase 表格之前，也需要先禁用表格。

3、Hive 常用操作

Hive 去年没有考操作题，但应该是在操作题的范围内的，所有在这里也简单总结一下吧。Hive 中的命令语法和 MySQL 还是比较类似的，如果你对 MySQL 比较数据可以直接略过啦~

创建表格

Hive是基于Hadoop的数据仓库解决方案，用于处理大数据集。要创建Hive表格，可以使用以下命令：

```
CREATE TABLE table_name (
  column1_name data_type,
  column2_name data_type,
  ...
)
[ROW FORMAT row_format]
[STORED AS file_format]
```

其中，`table_name` 是表格的名称，`column_name` 是表格中的列名，`data_type` 是列的数据类型。`ROW FORMAT` 和 `STORED AS` 部分是可选的。

例如，以下命令创建一个名为 `example_table` 的表格，其中包含两列 `id` 和 `name`：

```
CREATE TABLE example_table (
  id INT,
  name STRING
);
```

请注意，Hive不会立即创建表格。相反，它会将此命令解释为一个指令，并在将数据插入表格时动态地创建表格。

查看表格

查看表格结构：

```
DESCRIBE table_name;
```

查看表格数据：

```
SELECT * FROM table_name;
```

加载数据到表格

```
LOAD DATA [LOCAL] INPATH 'file_path' [OVERWRITE] INTO TABLE table_name;
```

插入数据到表格

```
INSERT INTO TABLE table_name VALUES (value1, value2, ...);
```

删除表格

```
DROP TABLE table_name;
```

修改表格结构

```
ALTER TABLE table_name ADD|DROP|RENAME column_name;
```

其他复杂操作

在Hive中，由于数据通常以文件格式存储在Hadoop分布式文件系统中，因此不支持原生的UPDATE和DELETE操作。因此，其他关系型数据库中常见的更新与删除操作需要创建一个新的表格来替代，比较复杂，应该不会考的，这里简单整理一下吧，感兴趣可以看一下。

1. 替代UPDATE操作：使用INSERT和SELECT

如果要更新表格中的数据，可以使用INSERT和SELECT语句的组合来实现。例如，以下语句将更新名为`example_table`的表格中`id`为1的记录中的`name`列：

```
INSERT OVERWRITE TABLE example_table
SELECT id, 'new_name' AS name
FROM example_table
WHERE id = 1;
```

这将创建一个新的表格，其中包含更新后的记录。原始表格不会被修改或删除。

2. 替代DELETE操作：使用CTAS和DROP

如果要删除表格中的某些数据，可以使用CREATE TABLE AS SELECT (CTAS)语句和DROP命令的组合来实现。例如，以下语句将删除名为`example_table`的表格中`id`为1的记录：

```
CREATE TABLE new_table AS
SELECT * FROM example_table
WHERE id <> 1;

DROP TABLE example_table;

ALTER TABLE new_table RENAME TO example_table;
```

这将创建一个新的表格`new_table`，其中不包含`id`为1的记录。然后，原始表格`example_table`将被删除，并且新表格将被重命名为`example_table`，以替换原始表格。

还有操作在其他课程中可能经常考到，但大数据大概率不会考的，也是感兴趣的话可以看一下。

创建视图：

```
CREATE VIEW view_name AS SELECT column1, column2, ... FROM table_name WHERE
condition;
```

聚合查询：

```
SELECT column1, function(column2) FROM table_name WHERE condition GROUP BY
column1;
```

连接查询：

```sql
SELECT table1.column1, table2.column2 FROM table1 JOIN table2 ON table1.column3
= table2.column4;
```