Hive DDL 实战工作手册（第 4 章）

《Hive 数据仓库应用》— 数据定义语言实习

工作手册目录

编号	任务内容	建议用时
准备	实验环境确认与数据集准备	20 分钟
任务4-1	HiveQL 操作数据库	30 分钟
任务4-2	HiveQL 操作数据表（内部表/外部表）	60 分钟
任务4-3	HiveQL 操作分区表	45 分钟
任务4-4	HiveQL 操作分桶表	35 分钟
任务4-5	HiveQL 操作视图	30 分钟
任务4-6	HiveQL 综合DDL操作（索引与Show语法）	30 分钟
综合练习	综合练习题	30 分钟
合计		**约4小时**

实验环境说明

基础环境

- 操作系统：Ubuntu Linux（VMware虚拟机）
- Hive 客户端工具：beeline（别名 `hive-bee`）
- Hive 数据库路径：`/user/hive/warehouse/unit4db`
- 数据文件目录：`/usr/datas/unit4/`

连接方式

打开终端，输入以下命令启动 beeline：

```bash
```

```
hive-bee
```

> **说明：** `hive-bee` 是系统预置的 beeline 快捷命令，执行后进入 `beeline>` 提示符，表示连接成功。

MySQL 信息（如需使用）

- 用户名：`hiveuser`
- 密码：`123456`
- 数据库名：`unit4db`

数据集文件路径

所有实习数据文件存放在：`/usr/datas/unit4/`

文件名	描述
`employees.txt`	员工信息数据
`products.txt`	产品信息数据
`sales.txt`	销售记录数据（用于分区表）
`orders.txt`	订单数据（用于分桶表）

准备工作：数据集创建（20 分钟）

在进行实习之前，需要先在Linux系统中准备好数据文件，并在Hive中创建实习数据库。

步骤 1：创建数据目录

打开 Linux 终端，执行以下命令创建数据目录：

```bash
sudo mkdir -p /usr/datas/unit4
sudo chmod 777 /usr/datas/unit4
```

步骤 2：创建数据文件

（1）创建员工数据文件 `employees.txt`

字段说明：`员工ID，姓名，部门，薪资，入职日期`

```bash
cat > /usr/datas/unit4/employees.txt << 'EOF'
```

```
1001,张伟,技术部,15000,2020-03-15
1002,李芳,市场部,12000,2019-07-20
1003,王磊,技术部,18000,2018-05-10
1004,赵敏,人事部,10000,2021-01-08
1005,陈浩,技术部,20000,2017-11-25
1006,刘洋,市场部,13500,2020-09-01
1007,杨静,财务部,11000,2022-02-14
1008,周强,技术部,16500,2019-04-30
1009,吴雪,市场部,12500,2021-06-18
1010,孙明,人事部,10500,2022-08-22
EOF
```

（2）创建产品数据文件 `products.txt`

字段说明：`产品ID，产品名称，类别，价格，库存数量`

```BASH
cat > /usr/datas/unit4/products.txt << 'EOF'
P001,笔记本电脑,电子产品,6999.00,150
P002,无线鼠标,电子产品,99.00,500
P003,机械键盘,电子产品,299.00,320
P004,办公椅,办公家具,1299.00,80
P005,会议桌,办公家具,3500.00,20
P006,文件柜,办公家具,899.00,60
P007,打印纸,办公耗材,45.00,1000
P008,中性笔,办公耗材,2.50,5000
P009,订书机,办公耗材,35.00,200
P010,投影仪,电子产品,4500.00,30
EOF
```

（3）创建销售数据文件 `sales.txt`（用于分区表）

字段说明：`销售ID，员工ID，产品ID，销售金额，销售日期`

```BASH
cat > /usr/datas/unit4/sales.txt << 'EOF'
S001,1001,P001,6999.00,2023-01-15
S002,1002,P007,45.00,2023-01-20
S003,1003,P002,99.00,2023-02-05
S004,1001,P010,4500.00,2023-02-18
S005,1005,P003,299.00,2023-03-10
S006,1006,P004,1299.00,2023-03-22
S007,1002,P008,2.50,2023-04-08
S008,1004,P005,3500.00,2023-04-15
S009,1003,P006,899.00,2023-05-02
S010,1007,P001,6999.00,2023-05-20
S011,1001,P002,99.00,2023-06-11
S012,1008,P009,35.00,2023-06-25
EOF
```

（4）创建订单数据文件 `orders.txt`（用于分桶表）

字段说明：`订单ID，客户名称，省份，产品类别，订单金额`

```BASH
cat > /usr/datas/unit4/orders.txt << 'EOF'
O001,北京科技有限公司,北京,电子产品,25000.00
O002,上海贸易公司,上海,办公家具,18000.00
O003,广州制造企业,广东,电子产品,32000.00
O004,深圳创新公司,广东,办公耗材,5000.00
O005,北京教育机构,北京,电子产品,45000.00
O006,上海金融公司,上海,办公家具,22000.00
O007,广州零售连锁,广东,办公耗材,8000.00
O008,北京医疗机构,北京,电子产品,38000.00
O009,杭州互联网公司,浙江,电子产品,55000.00
O010,南京工程公司,江苏,办公家具,12000.00
O011,武汉大学,湖北,电子产品,28000.00
O012,成都科技园区,四川,电子产品,42000.00
O013,杭州电商平台,浙江,办公耗材,6500.00
O014,苏州制造业,江苏,办公家具,15000.00
O015,武汉研究院,湖北,电子产品,33000.00
EOF
```

步骤 3：验证文件创建是否成功

```BASH
ls -la /usr/datas/unit4/
cat /usr/datas/unit4/employees.txt
```

> **检查点：** 确认4个数据文件均已创建，且内容正确后，再进行后续实习。

步骤 4：创建 Hive 实习数据库

启动beeline并创建数据库：

```BASH
hive-bee
```

在 beeline 中执行：

```SQL
-- 创建实习数据库
CREATE DATABASE IF NOT EXISTS unit4db
COMMENT '第4章DDL实习数据库'
LOCATION '/user/hive/warehouse/unit4db';
```

```
-- 确认数据库创建成功
SHOW DATABASES;


-- 切换到unit4db数据库
USE unit4db;
```

任务 4-1：HiveQL 操作数据库（30 分钟）

学习目标

- 掌握 Hive 数据库的创建、查询、查看、切换、修改、删除操作
- 熟悉 CREATE DATABASE、SHOW DATABASES、DESC DATABASE、USE、ALTER DATABASE、DROP DATABASE 语法

实习 4-1-1：创建数据库

任务描述： 在 Hive 中创建一个用于本章实习的数据库，并指定 HDFS 存储路径和描述信息。

操作步骤：

```sql
-- 1. 创建基本数据库
CREATE DATABASE IF NOT EXISTS unit4db
COMMENT '第4章HiveQL DDL实习专用数据库'
LOCATION '/user/hive/warehouse/unit4db';
```

```sql
-- 2. 创建带有自定义属性的数据库（练习）
CREATE DATABASE IF NOT EXISTS unit4db_test
COMMENT '测试用数据库'
WITH DBPROPERTIES (
  'creator' = 'student',
  'create_date' = '2024-01-01',
  'version' = '1.0'
);
```

> **语法要点：**
> - `IF NOT EXISTS`：防止数据库已存在时报错
> - `COMMENT`：为数据库添加描述
> - `LOCATION`：指定 HDFS 上的存储路径

- `WITH DBPROPERTIES`：设置自定义属性（键值对）

实习 4-1-2：查询数据库

任务描述： 查看当前 Hive 中所有数据库，并使用模糊查询筛选特定数据库。

```sql
-- 1. 查看所有数据库
SHOW DATABASES;
```

```sql
-- 2. 使用模糊查询：查询以 "unit" 开头的数据库
SHOW DATABASES LIKE 'unit*';
```

```sql
-- 3. 使用模糊查询：查询包含 "4" 的数据库
SHOW DATABASES LIKE '*4*';
```

语法要点：

- `LIKE`子句支持通配符 `*`（匹配任意字符）和 `|`（逻辑或）

实习 4-1-3：查看数据库详细信息

任务描述： 查看 unit4db 数据库的详细信息，包括描述、存储路径和属性。

```sql
-- 1. 查看数据库基本信息
DESC DATABASE unit4db;
```

```sql
-- 2. 查看数据库扩展信息（包含属性）
DESC DATABASE EXTENDED unit4db;
```

```sql
-- 3. 查看带属性的测试数据库详细信息
DESC DATABASE EXTENDED unit4db_test;
```

观察： 对比 `DESC DATABASE` 和 `DESC DATABASE EXTENDED` 的输出结果，注意 EXTENDED 关键字会显示通过 `WITH DBPROPERTIES` 设置的额外属性。

实习 4-1-4：切换数据库

```sql
-- 切换到 unit4db 数据库
USE unit4db;


-- 验证当前数据库
SELECT current_database();
```

实习 4-1-5：修改数据库

任务描述： 修改 unit4db_test 数据库的属性信息。

```sql
-- 修改数据库属性
ALTER DATABASE unit4db_test
SET DBPROPERTIES (
  'last_modified' = '2024-06-01',
  'status' = 'active'
);


-- 验证修改结果
DESC DATABASE EXTENDED unit4db_test;
```

实习 4-1-6：删除数据库

```sql
-- 删除测试数据库（注意：只能删除空数据库）
DROP DATABASE IF EXISTS unit4db_test;


-- 确认删除成功
SHOW DATABASES;
```

> **注意：** 如果数据库中存在表，需要使用 `CASCADE` 选项：
> sql
> DROP DATABASE IF EXISTS unit4db_test CASCADE;
> ```


练习题 4-1

练习1： 创建一个名为 `practice_db` 的数据库，要求：

- 添加注释："练习用数据库"
- 指定 HDFS 路径为：`/user/hive/warehouse/practice_db`
- 添加属性：`author=学生姓名`，`subject=大数据技术`

练习2： 使用 LIKE 子句，查询所有以字母 "p" 开头的数据库。

练习3： 查看 `practice_db` 的详细信息，确认属性是否正确设置。

练习4： 为 `practice_db` 增加新属性 `semester=2024-spring`，并验证修改结果。

练习5： 删除 `practice_db` 数据库。

任务 4-2：HiveQL 操作数据表（60 分钟）

学习目标

- 掌握内部表（Managed Table）和外部表（External Table）的创建
- 掌握数据表的查看、修改、删除操作
- 理解内部表和外部表的区别

实习 4-2-1：创建内部表（Managed Table）

任务描述： 在 unit4db 数据库中创建员工信息内部表 `employees`。

操作前确认当前数据库：

```sql
USE unit4db;
SELECT current_database();
```

创建内部表：

```sql
CREATE TABLE IF NOT EXISTS unit4db.employees (
  emp_id     INT         COMMENT '员工ID',
  emp_name   STRING      COMMENT '员工姓名',
  dept       STRING      COMMENT '所属部门',
  salary     DOUBLE      COMMENT '薪资',
  hire_date  STRING      COMMENT '入职日期'
)
COMMENT '员工信息内部表'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
```

```
STORED AS TEXTFILE
TBLPROPERTIES ('creator'='student', 'table_type'='managed');
```

> **语法要点：**
> - `ROW FORMAT DELIMITED FIELDS TERMINATED BY ','`：指定字段分隔符为逗号
> - `LINES TERMINATED BY '\n'`：指定行分隔符为换行符
> - `STORED AS TEXTFILE`：以文本格式存储
> - `TBLPROPERTIES`：设置表属性

实习 4-2-2：创建外部表（External Table）

任务描述： 在 unit4db 数据库中创建产品信息外部表 `products_ext`，并指定外部存储路径。

```sql
CREATE EXTERNAL TABLE IF NOT EXISTS unit4db.products_ext (
  prod_id    STRING      COMMENT '产品ID',
  prod_name  STRING      COMMENT '产品名称',
  category   STRING      COMMENT '产品类别',
  price      DOUBLE      COMMENT '价格',
  stock      INT         COMMENT '库存数量'
)
COMMENT '产品信息外部表'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/user/hive/warehouse/unit4db/products_ext'
TBLPROPERTIES ('creator'='student', 'table_type'='external');
```

> **内外部表对比：**
> | 特性 | 内部表 | 外部表 |
> |---|---|---|
> | 关键字 | 无（默认） | `EXTERNAL` |
> | 删除表时 | 元数据+数据文件均删除 | 只删除元数据，数据文件保留 |
> | 适用场景 | Hive独占管理数据 | 多系统共享数据 |

实习 4-2-3：向表中加载数据

任务描述： 将本地文件中的数据加载到 Hive 表中。

```sql
-- 将本地 employees.txt 数据加载到内部表 employees
```

```
LOAD DATA LOCAL INPATH '/usr/datas/unit4/employees.txt'
INTO TABLE unit4db.employees;
```

```SQL
-- 将本地 products.txt 数据加载到外部表 products_ext
LOAD DATA LOCAL INPATH '/usr/datas/unit4/products.txt'
INTO TABLE unit4db.products_ext;
```

```SQL
-- 验证数据加载成功
SELECT * FROM unit4db.employees;
SELECT COUNT(*) FROM unit4db.employees;
```

> **关键注意：** 使用 `LOCAL` 关键字表示从本地文件系统加载，不加 `LOCAL` 则从 HDFS 加载。

实习 4-2-4：查看数据表信息

```SQL
-- 1. 查看 unit4db 中所有数据表
USE unit4db;
SHOW TABLES;
```

```SQL
-- 2. 模糊查询：查看以 "e" 开头的数据表
SHOW TABLES LIKE 'e*';
```

```SQL
-- 3. 查看内部表 employees 的基本结构信息
DESC employees;
```

```SQL
-- 4. 查看外部表 products_ext 的详细结构信息（含存储格式、文件路径等）
DESC FORMATTED products_ext;
```

> **观察重点：** 执行 `DESC FORMATTED` 后，注意查看以下字段：
> - `Table Type`：判断是内部表（MANAGED_TABLE）还是外部表（EXTERNAL_TABLE）
> - `Location`：表数据在 HDFS 上的存储路径

实习 4-2-5：修改数据表结构

(1) 创建练习用表

```sql
-- 创建结构修改练习表
CREATE TABLE IF NOT EXISTS unit4db.alter_test (
  id     STRING  COMMENT '编号',
  gender STRING  COMMENT '性别',
  name   STRING  COMMENT '姓名'
);
```

(2) 重命名表

```sql
-- 将 alter_test 重命名为 alter_test_new
ALTER TABLE alter_test RENAME TO alter_test_new;


-- 验证重命名结果
SHOW TABLES;
```

(3) 修改列定义（重命名列、移动位置、修改注释）

```sql
-- 将 gender 列改名为 sex，并移动到 name 列之后，同时更新注释
ALTER TABLE alter_test_new
CHANGE gender sex STRING
COMMENT '性别字段'
AFTER name;


-- 查看修改结果
DESC alter_test_new;
```

(4) 修改列的数据类型

```sql
-- 将 sex 列的数据类型由 STRING 改为 VARCHAR(10)
ALTER TABLE alter_test_new
CHANGE sex sex VARCHAR(10)
COMMENT '性别字段';


-- 查看修改结果
DESC alter_test_new;
```

(5) 添加新列

```sql
-- 在表末尾添加 age 和 email 两列
```

```
ALTER TABLE alter_test_new
ADD COLUMNS (
  age   INT    COMMENT '年龄',
  email STRING COMMENT '电子邮件'
);


-- 查看修改结果
DESC alter_test_new;
```

(6) 替换所有列（重置表结构）

```SQL
-- 替换表中所有列（谨慎使用！）
ALTER TABLE alter_test_new
REPLACE COLUMNS (
  user_id   INT    COMMENT '用户ID',
  user_name STRING COMMENT '用户名',
  password  STRING COMMENT '密码'
);


-- 查看修改结果
DESC alter_test_new;
```

(7) 修改表属性

```SQL
ALTER TABLE alter_test_new
SET TBLPROPERTIES ('last_modified' = '2024-06-01', 'status' = 'test');


-- 查看表详细信息
DESC FORMATTED alter_test_new;
```

实习 4-2-6：删除数据表

(1) 删除练习表

```SQL
-- 删除 alter_test_new 表（PURGE 表示不经过回收站，直接删除）
DROP TABLE IF EXISTS alter_test_new PURGE;


-- 验证删除成功
SHOW TABLES;
```

> **思考：** 删除内部表 `employees` 和外部表 `products_ext` 后，HDFS 上的数据文件会发生什么变化？

（2）验证内外部表删除的区别（观察实验）

```sql
-- 查看内部表数据文件路径（删除前记录路径）
DESC FORMATTED employees;


-- 查看外部表数据文件路径（删除前记录路径）
DESC FORMATTED products_ext;
```

```bash
# 在Linux终端检查HDFS上的文件（删除表后再检查对比）
hdfs dfs -ls /user/hive/warehouse/unit4db/
```

练习题 4-2

练习1： 创建内部表 `dept_info`，字段如下：

- `dept_id`：INT，部门编号
- `dept_name`：STRING，部门名称
- `manager`：STRING，部门经理
- `location`：STRING，办公地点
- 字段分隔符设置为制表符 `\t`

练习2： 创建外部表 `salary_grade_ext`，字段如下：

- `grade_id`：INT，薪资等级
- `min_salary`：DOUBLE，最低薪资
- `max_salary`：DOUBLE，最高薪资
- `grade_name`：STRING，等级名称
- 指定 LOCATION 为 `/user/hive/warehouse/unit4db/salary_grade`

练习3： 向 `dept_info` 添加两个新列：`phone`（STRING，联系电话）和 `email`（STRING，部门邮箱），然后用 DESC 命令验证。

练习4： 将 `dept_info` 表的 `manager` 列重命名为 `dept_manager`，并将其移动到 `dept_name` 列之后。

练习5： 使用 `DESC FORMATTED` 命令，分别查看 `dept_info`（内部表）和 `salary_grade_ext`（外部表）的详细信息，记录 `Table Type` 字段的值，思考两者的区别。

任务 4-3：HiveQL 操作分区表（45 分钟）

学习目标

- 理解 Hive 分区表的概念和作用
- 掌握分区表的创建、查看、添加、重命名、移动和删除操作
- 理解静态分区的使用方式

分区表概念回顾

> **为什么需要分区表?**
> - Hive 默认进行全表扫描，当数据量很大时效率低下
> - 分区表将数据按分区字段分成多个子目录，查询时只扫描指定分区
> - 例如：将销售数据按年份和月份分区，查询特定月份时只扫描对应分区目录

实习 4-3-1：创建分区表

任务描述： 创建销售数据分区表，以年份和月份作为分区字段。

```SQL
-- 创建二级分区表（按年份和月份分区）
CREATE TABLE IF NOT EXISTS unit4db.sales_partitioned (
  sale_id   STRING COMMENT '销售记录ID',
  emp_id    INT    COMMENT '员工ID',
  prod_id   STRING COMMENT '产品ID',
  amount    DOUBLE COMMENT '销售金额',
  sale_date STRING COMMENT '销售日期'
)
COMMENT '销售数据分区表'
PARTITIONED BY (
  sale_year  STRING COMMENT '销售年份',
  sale_month STRING COMMENT '销售月份'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

> **重要注意事项：**
> 1. 分区字段名称（`sale_year`, `sale_month`）不能与表的普通列名重复
> 2. 分区字段一旦创建，无法修改或添加

实习 4-3-2：查看分区信息

```sql
-- 查看 sales_partitioned 表的所有分区（初始状态为空）
SHOW PARTITIONS unit4db.sales_partitioned;


-- 查看表的详细信息
DESC FORMATTED unit4db.sales_partitioned;
```

实习 4-3-3：添加分区

（1）手动添加空分区

```sql
-- 添加 2023年1月 分区
ALTER TABLE unit4db.sales_partitioned
ADD PARTITION (sale_year='2023', sale_month='01')
LOCATION '/user/hive/warehouse/unit4db/sales_partitioned/year=2023/month=01';


-- 添加 2023年2月 分区
ALTER TABLE unit4db.sales_partitioned
ADD PARTITION (sale_year='2023', sale_month='02')
LOCATION '/user/hive/warehouse/unit4db/sales_partitioned/year=2023/month=02';


-- 同时添加多个分区
ALTER TABLE unit4db.sales_partitioned
ADD IF NOT EXISTS
PARTITION (sale_year='2023', sale_month='03')
PARTITION (sale_year='2023', sale_month='04')
PARTITION (sale_year='2023', sale_month='05')
PARTITION (sale_year='2023', sale_month='06');
```

```sql
-- 确认分区添加成功
SHOW PARTITIONS unit4db.sales_partitioned;
```

（2）向分区加载数据

首先，准备分月份数据文件：

```bash
# 在Linux终端创建按月份分割的数据文件
grep "2023-01" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_01.txt
```

```
grep "2023-02" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_02.txt
grep "2023-03" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_03.txt
grep "2023-04" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_04.txt
grep "2023-05" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_05.txt
grep "2023-06" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_06.txt
```

在 beeline 中加载数据：

SQL

```
-- 向指定分区加载数据
LOAD DATA LOCAL INPATH '/usr/datas/unit4/sales_2023_01.txt'
INTO TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='01');


LOAD DATA LOCAL INPATH '/usr/datas/unit4/sales_2023_02.txt'
INTO TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='02');


LOAD DATA LOCAL INPATH '/usr/datas/unit4/sales_2023_03.txt'
INTO TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='03');
```

SQL

```
-- 验证分区数据
SELECT * FROM unit4db.sales_partitioned WHERE sale_year='2023' AND sale_month='01';
SELECT COUNT(*) FROM unit4db.sales_partitioned;
```

实习 4-3-4：重命名分区

SQL

```
-- 将 2023年01月 分区重命名为 2023年1月（格式统一）
ALTER TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='01')
RENAME TO PARTITION (sale_year='2023', sale_month='1');


-- 验证重命名结果
SHOW PARTITIONS unit4db.sales_partitioned;
```

实习 4-3-5：创建第二张分区表并移动分区

（1）创建结构相同的分区表 `sales_archive`

```sql
CREATE TABLE IF NOT EXISTS unit4db.sales_archive (
  sale_id   STRING COMMENT '销售记录ID',
  emp_id    INT    COMMENT '员工ID',
  prod_id   STRING COMMENT '产品ID',
  amount    DOUBLE COMMENT '销售金额',
  sale_date STRING COMMENT '销售日期'
)
COMMENT '销售归档分区表'
PARTITIONED BY (
  sale_year  STRING COMMENT '销售年份',
  sale_month STRING COMMENT '销售月份'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

(2) 移动分区（将 sales_partitioned 的某分区移动到 sales_archive)

```sql
-- 将 2023年6月分区从 sales_partitioned 移动到 sales_archive
ALTER TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='06')
RENAME TO PARTITION (sale_year='2023', sale_month='06');


-- 注意：实际移动操作需通过 exchange partition 语法
ALTER TABLE unit4db.sales_archive
EXCHANGE PARTITION (sale_year='2023', sale_month='06')
WITH TABLE unit4db.sales_partitioned;


-- 验证移动结果
SHOW PARTITIONS unit4db.sales_partitioned;
SHOW PARTITIONS unit4db.sales_archive;
```

实习 4-3-6：删除分区

```sql
-- 删除 sales_partitioned 中的 2023年04月 分区
ALTER TABLE unit4db.sales_partitioned
DROP IF EXISTS PARTITION (sale_year='2023', sale_month='04');


-- 验证删除结果
SHOW PARTITIONS unit4db.sales_partitioned;
```

> **思考：** 删除分区后，HDFS 上对应目录中的数据文件是否还存在？

练习题 4-3

练习1： 创建员工出勤记录分区表 `attendance`，字段包括：

- `att_id`（INT），`emp_id`（INT），`att_status`（STRING，出勤状态：出勤/请假/出差）
- 分区字段：`att_year`（STRING），`att_month`（STRING），`att_day`（STRING）

练习2： 为 `attendance` 表添加以下分区，并验证：

- 分区1：`att_year='2024'`, `att_month='01'`, `att_day='15'`
- 分区2：`att_year='2024'`, `att_month='01'`, `att_day='16'`
- 分区3：`att_year='2024'`, `att_month='02'`, `att_day='01'`

练习3： 查询 `attendance` 表，只查询 `att_year='2024'` 且 `att_month='01'` 的所有分区信息。

```sql
SHOW PARTITIONS attendance PARTITION(att_year='2024', att_month='01');
```

练习4： 删除 `att_year='2024'`, `att_month='02'` 的分区，并验证删除结果。

任务 4-4：HiveQL 操作分桶表（35 分钟）

学习目标

- 理解分桶表的概念和用途
- 掌握分桶表的创建和数据加载方法
- 了解分桶在数据倾斜场景下的优化作用

分桶表概念回顾

> **分桶 vs 分区：**
> - **分区**：按某个字段的值将数据存储到不同目录
> - **分桶**：按某个字段的哈希值将数据均匀分配到固定数量的文件（桶）中
> - **解决问题**：分桶解决分区可能带来的数据倾斜问题

实习 4-4-1：创建分桶表

任务描述： 创建订单分桶表，根据省份字段将数据分成4个桶。

步骤1：开启强制分桶配置

```SQL
-- 开启强制分桶（Hive 3.x 通常默认支持，但建议明确设置）
SET hive.enforce.bucketing = true;
```

步骤2：创建分桶表

```SQL
CREATE TABLE IF NOT EXISTS unit4db.orders_bucketed (
  order_id  STRING COMMENT '订单ID',
  cust_name STRING COMMENT '客户名称',
  province  STRING COMMENT '省份',
  category  STRING COMMENT '产品类别',
  amount    DOUBLE COMMENT '订单金额'
)
COMMENT '订单分桶表（按省份分4桶）'
CLUSTERED BY (province) INTO 4 BUCKETS
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

```SQL
-- 创建带排序的分桶表（分桶内按金额降序排列）
CREATE TABLE IF NOT EXISTS unit4db.orders_bucketed_sorted (
  order_id  STRING COMMENT '订单ID',
  cust_name STRING COMMENT '客户名称',
  province  STRING COMMENT '省份',
  category  STRING COMMENT '产品类别',
  amount    DOUBLE COMMENT '订单金额'
)
COMMENT '订单分桶排序表'
CLUSTERED BY (province) SORTED BY (amount DESC) INTO 4 BUCKETS
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

> **分桶规则说明：**
> - 分桶字段的哈希值对桶数取模，结果相同的数据进入同一个桶
> - `INT` 类型：`hash_function(int) = int 本身`
> - 其他类型（STRING等）：使用 Java `hashCode()` 方法

实习 4-4-2：向分桶表加载数据

> **重要：** 分桶表不能直接用 `LOAD DATA` 加载，必须通过 INSERT INTO ... SELECT 方式，或先创建临时表再 INSERT。

步骤1：创建临时中间表

```sql
CREATE TABLE IF NOT EXISTS unit4db.orders_temp (
  order_id  STRING,
  cust_name STRING,
  province  STRING,
  category  STRING,
  amount    DOUBLE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

步骤2：将数据加载到临时表

```sql
LOAD DATA LOCAL INPATH '/usr/datas/unit4/orders.txt'
INTO TABLE unit4db.orders_temp;


-- 验证数据加载
SELECT * FROM unit4db.orders_temp;
```

步骤3：通过 INSERT INTO ... SELECT 将数据写入分桶表

```sql
-- 开启MapReduce任务（某些Hive版本需要此设置）
SET hive.execution.engine = mr;


INSERT INTO TABLE unit4db.orders_bucketed
SELECT order_id, cust_name, province, category, amount
FROM unit4db.orders_temp;
```

步骤4：验证分桶效果

```sql
-- 查看分桶表数据
SELECT * FROM unit4db.orders_bucketed;


-- 按省份统计（观察各桶分配情况）
SELECT province, COUNT(*) AS cnt FROM unit4db.orders_bucketed GROUP BY province;
```

```bash
# 在Linux终端查看HDFS上的分桶文件（应有4个文件）
hdfs dfs -ls /user/hive/warehouse/unit4db/orders_bucketed/
```

实习 4-4-3：查看分桶表信息

```sql
-- 查看分桶表的详细元数据信息
DESC FORMATTED unit4db.orders_bucketed;
```

> **观察：** 在 DESC FORMATTED 的输出中，找到 `Num Buckets`（桶数）和 `Bucket Columns`（分桶字段），确认与建表时设置一致。

练习题 4-4

练习1： 创建员工薪资分桶表 `emp_salary_bucketed`，要求：

- 字段：`emp_id`（INT），`emp_name`（STRING），`dept`（STRING），`salary`（DOUBLE）
- 按 `dept`（部门）分为 **3 个桶**
- 桶内按 `salary` 升序排列

练习2： 先创建临时表 `emp_salary_temp` 加载 `employees.txt` 数据，然后通过 INSERT INTO ... SELECT 将数据写入 `emp_salary_bucketed`。

练习3： 通过 `DESC FORMATTED` 验证分桶表的 `Num Buckets` 为 3，`Bucket Columns` 为 `dept`。

练习4： 执行查询，统计每个部门的员工数量，观察分桶是否均匀。

```sql
SELECT dept, COUNT(*) AS emp_count FROM emp_salary_bucketed GROUP BY dept;
```

任务 4-5：HiveQL 操作视图（30 分钟）

学习目标

- 理解 Hive 视图的概念和特点
- 掌握视图的创建、查看、修改和删除操作

视图概念回顾

> **Hive 视图特点：**
> - 视图是虚拟表，只保存查询定义，不存储数据
> - 视图可以简化复杂查询语句
> - 视图不提高查询性能，也不缓存数据
> - 基础表结构变更（删除/修改）可能导致视图失效

实习 4-5-1：创建视图

任务描述： 基于 `employees` 表创建技术部员工视图。

```SQL
-- 确保 employees 表有数据
USE unit4db;
SELECT COUNT(*) FROM employees;
```

（1）创建简单视图

```SQL
-- 创建技术部员工视图
CREATE VIEW IF NOT EXISTS unit4db.v_tech_employees
AS
SELECT emp_id, emp_name, salary, hire_date
FROM unit4db.employees
WHERE dept = '技术部';
```

（2）创建包含列注释的视图

```SQL
-- 创建高薪员工视图（薪资>15000），并为列添加注释
CREATE VIEW IF NOT EXISTS unit4db.v_high_salary_emp
(
  employee_id   COMMENT '员工编号',
  employee_name COMMENT '员工姓名',
  department    COMMENT '所在部门',
  monthly_salary COMMENT '月薪'
)
COMMENT '月薪超过15000元的员工视图'
AS
SELECT emp_id, emp_name, dept, salary
FROM unit4db.employees
WHERE salary > 15000;
```

（3）创建聚合统计视图

```SQL
```

```
-- 创建部门薪资统计视图
CREATE VIEW IF NOT EXISTS unit4db.v_dept_salary_stats
COMMENT '部门薪资统计视图'
AS
SELECT
  dept              AS department,
  COUNT(emp_id)     AS emp_count,
  AVG(salary)       AS avg_salary,
  MAX(salary)       AS max_salary,
  MIN(salary)       AS min_salary
FROM unit4db.employees
GROUP BY dept;
```

实习 4-5-2：查看视图

SQL

```
-- 1. 查看当前数据库中所有视图
SHOW VIEWS IN unit4db;
```

SQL

```
-- 2. 模糊查询：查看以 "v_" 开头的视图
SHOW VIEWS IN unit4db LIKE 'v_*';
```

SQL

```
-- 3. 查看视图结构基本信息
DESC unit4db.v_tech_employees;
```

SQL

```
-- 4. 查看视图详细信息（含视图定义语句）
DESC FORMATTED unit4db.v_high_salary_emp;
```

实习 4-5-3：查询视图数据

SQL

```
-- 查询技术部员工视图
SELECT * FROM unit4db.v_tech_employees;
```

SQL

```
-- 查询高薪员工视图，并按薪资降序排列
SELECT * FROM unit4db.v_high_salary_emp
```

```
ORDER BY monthly_salary DESC;
```

SQL

```
-- 查询部门统计视图
SELECT * FROM unit4db.v_dept_salary_stats;
```

实习 4-5-4：修改视图

(1) 修改视图属性

SQL

```
ALTER VIEW unit4db.v_tech_employees
SET TBLPROPERTIES ('last_modified' = '2024-06-01', 'status' = 'active');
```

(2) 修改视图的查询定义

SQL

```
-- 修改 v_tech_employees 视图：包含所有部门员工，不再限制技术部
ALTER VIEW unit4db.v_tech_employees
AS
SELECT emp_id, emp_name, dept, salary, hire_date
FROM unit4db.employees
WHERE salary >= 12000;


-- 验证修改结果
SELECT * FROM unit4db.v_tech_employees;
DESC FORMATTED unit4db.v_tech_employees;
```

实习 4-5-5：删除视图

SQL

```
-- 删除视图
DROP VIEW IF EXISTS unit4db.v_tech_employees;
DROP VIEW IF EXISTS unit4db.v_high_salary_emp;


-- 验证删除结果
SHOW VIEWS IN unit4db;
```

练习题 4-5

练习1： 基于 `employees` 表，创建视图 `v_dept_emp_list`，显示各部门员工列表，字段包括：部门名称、员工ID、员工姓名、薪资，并按部门名称升序、薪资降序排列。

练习2： 基于 `orders_bucketed` 表，创建视图 `v_province_orders`，统计各省份的订单数量和总金额。

练习3： 修改视图 `v_dept_emp_list`，增加过滤条件：只显示薪资高于部门平均薪资的员工。

练习4： 使用 `DESC FORMATTED` 查看 `v_dept_emp_list` 的视图定义，确认查询语句是否正确。

任务 4-6：HiveQL 综合 DDL 操作（30 分钟）

学习目标

- 复习和巩固 Hive DDL 操作（数据库、表、分区、视图）
- 了解 Hive 索引的基本概念和操作
- 掌握 SHOW 系列语法的综合使用

实习 4-6-1：SHOW 语法综合应用

```SQL
-- 1. 显示所有数据库
SHOW DATABASES;


-- 2. 显示 unit4db 中所有表
USE unit4db;
SHOW TABLES;


-- 3. 显示 unit4db 中所有视图
SHOW VIEWS;


-- 4. 显示 sales_partitioned 表的分区信息
SHOW PARTITIONS sales_partitioned;


-- 5. 显示表的创建语句
SHOW CREATE TABLE employees;


-- 6. 显示表列信息
SHOW COLUMNS IN employees;


-- 7. 显示函数列表（了解用）
SHOW FUNCTIONS;
```

实习 4-6-2：Hive 索引操作（了解）

> **注意：** Hive 3.x 中已废弃索引功能，以下内容仅供了解，不建议在新版本中使用。在实际生产中，通常使用分区、ORC文件格式的内置索引或 Bloom Filter 来优化查询性能。

```SQL
-- 了解索引语法（Hive 2.x 及以下版本有效）
/*
CREATE INDEX idx_emp_name
ON TABLE employees (emp_name)
AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler'
WITH DEFERRED REBUILD
TBLPROPERTIES ('creator' = 'student');
*/


-- 在现代 Hive 3.x 中，优化查询性能的推荐方式：
-- 1. 使用分区表
-- 2. 使用 ORC 文件格式（内置轻量索引）
-- 3. 合理使用分桶


-- 演示：将 employees 表转换为 ORC 格式（更好的查询性能）
CREATE TABLE IF NOT EXISTS unit4db.employees_orc (
  emp_id    INT     COMMENT '员工ID',
  emp_name  STRING  COMMENT '员工姓名',
  dept      STRING  COMMENT '所属部门',
  salary    DOUBLE  COMMENT '薪资',
  hire_date STRING  COMMENT '入职日期'
)
STORED AS ORC
TBLPROPERTIES ('orc.compress'='SNAPPY');


-- 从 employees 表插入数据到 ORC 表
INSERT INTO unit4db.employees_orc
SELECT * FROM unit4db.employees;


-- 验证
SELECT * FROM unit4db.employees_orc;
```

实习 4-6-3：综合 DDL 操作回顾

```SQL
-- 1. 查看本次实习创建的所有对象
```

```
USE unit4db;
SHOW TABLES;
SHOW VIEWS;


-- 2. 查看各表的行数
SELECT 'employees' AS table_name, COUNT(*) AS row_count FROM employees
UNION ALL
SELECT 'products_ext', COUNT(*) FROM products_ext
UNION ALL
SELECT 'orders_bucketed', COUNT(*) FROM orders_bucketed
UNION ALL
SELECT 'sales_partitioned', COUNT(*) FROM sales_partitioned;
```

综合练习题

综合练习（30分钟）

按照以下要求，完成一个完整的员工管理数据库的 DDL 操作：

要求1：数据库操作

- 创建数据库 `hr_db`，注释为"人力资源数据库"，HDFS路径为 `/user/hive/warehouse/hr_db`
- 添加属性：`dept=HR`, `version=2.0`
- 修改属性：添加 `last_update=2024-06-01`

要求2：表操作

- 在 `hr_db` 中创建内部表 `hr_employees`，字段如下：

- `emp_id` INT, `emp_name` STRING, `age` INT, `dept_id` INT, `salary` DOUBLE, `city` STRING

- 字段分隔符为逗号

- 在 `hr_db` 中创建外部表 `hr_departments`，字段：`dept_id` INT, `dept_name` STRING, `dept_head` STRING

- LOCATION: `/user/hive/warehouse/hr_db/departments`

要求3：分区表

- 在 `hr_db` 中创建分区表 `hr_salary_history`，按 `year`（STRING）和 `quarter`（STRING）分区
- 字段：`emp_id` INT, `base_salary` DOUBLE, `bonus` DOUBLE
- 添加以下分区：year='2024',quarter='Q1' 和 year='2024',quarter='Q2'

要求4：分桶表

- 在 `hr_db` 中创建分桶表 `hr_employees_bucketed`
- 与 `hr_employees` 字段相同，按 `dept_id` 分为 4 个桶，桶内按 `salary` 降序排列

要求5：视图

- 创建视图 `v_senior_employees`：显示工资高于 12000 的员工信息（全字段）

- 创建视图 `v_city_stats`：统计各城市员工数量和平均薪资

要求6：查询验证

- 使用 SHOW 命令验证所有创建的对象
- 使用 DESC FORMATTED 查看 `hr_employees`（内部表）和 `hr_departments`（外部表）的类型区别

本章实习总结

通过本次实习，我们掌握了以下 Hive DDL 核心操作：

操作对象	关键命令
数据库	CREATE/SHOW/DESC/USE/ALTER/DROP DATABASE
数据表	CREATE TABLE（内部/外部）/SHOW TABLES/DESC/ALTER TABLE/DROP TABLE
分区表	PARTITIONED BY/ADD PARTITION/RENAME/EXCHANGE/DROP PARTITION
分桶表	CLUSTERED BY/SORTED BY/INTO N BUCKETS
视图	CREATE/SHOW/DESC/ALTER/DROP VIEW
综合	SHOW CREATE TABLE/SHOW COLUMNS/STORED AS ORC

数据加载方式：

```sql
-- 从本地文件系统加载
LOAD DATA LOCAL INPATH '/usr/datas/unit4/文件名.txt' INTO TABLE 表名;


-- 加载到指定分区
LOAD DATA LOCAL INPATH '/usr/datas/unit4/文件名.txt'
INTO TABLE 表名 PARTITION (分区字段='值');


-- 分桶表通过 INSERT INTO ... SELECT 加载
INSERT INTO TABLE 分桶表 SELECT * FROM 临时表;
```

本工作手册基于《Hive数据仓库应用教程》第4章内容编写，适用于约4小时的课堂实习。

使用环境：Hadoop 3.3.5 + Apache Hive 3.1.3 + VMware Ubuntu