```
-- 切换路径
cd /home/dmdba/dmdbms/tool

-- 查看当前工作目录
pwd

-- 查看当前目录信息
ls

-- 调用配置助手(tool 目录下)
./dbca.sh

-- 切换 root 账号
su -

-- 返回账号
EXIT

-- 打开数据库配置软件(tool 目录下)
./manager


-- 查看当前数据库状态
FROM v$instance

--切换到数据库所在目录
cd /home/dmdba/dmdbms/bin

-- 查看数据库服务名
ll DmS*

-- 查看数据库是否运行中（数据库服务名以 DMSERVER 为例）
--./DmService 数据库服务名  status
./DmServiceDMSERVER status
-- 关闭数据库
./DmServiceDMSERVER stop
-- 打开数据库
./DmServiceDMSERVER start
-- 图形化打开关闭数据库(tool 目录下)
./deservice.sh

-- 查询（以 DMDL 模式下 employee 表，用户 test 为例）
-- 全列查询
SELECT * FROM dmdl.employee;
```

```
-- 单列查询
SELECT employee_name FROM dmdl.employee;
-- 多列查询
SELECT employee_name,employee_id FROM dmdl.employee;
-- 条件查询
SELECT employee_name FROM dmdl.employee WHERE employee_id = 1;
-- 去重查询 DISTINCT
SELECT DISTINCT employee_id FROM dmdl.employee;

-- 对列进行命名 a
SELECT employee_name AS a FROM dmdl.employee;
--等价于
SELECT employee_name a FROM dmdl.employee;

-- 连接列(||)
SELECT employee_name || '的 ID 是' || employee_id FROM dmdl.employee;


-- 排序查询 order by（desc 降序，默认：asc 升序）注意放在 SQL 语句的句尾
-- 升序 数字类型：从小到大 -- 日期时间类型：从远到近
-- 降序 数字类型：从大到小 -- 日期时间类型：从近到远
SELECT employee_salary FROM dmdl.employee ORDER BY salary;
SELECT employee_salary FROM dmdl.employee ORDER BY salary desc;


-- 分组查询 group by
-- 分组函数，除聚合函数的列，不用跟在 group by 后面，其他列都要
COUNT 统计函数 select COUNT(*) FROM dmdl.employee;
SUM 求和 select SUM(salary) FROM dmdl.employee;
MAX 最大值 select MAX(salary) FROM dmdl.employee;
MIN 最小值 select MIN(salary) FROM dmdl.employee;
AVG 平均值 select AVG(salary) FROM dmdl.employee;
-- 求每个部门工资总和
select SUM(salary) FROM dmdl.employee GROUP BY department_id;

-- having 关键字，是对 group 的进一步过滤，没有 group by 时不能用
-- 查询部门工资总和在 15000 以上的,降序排序
select SUM(salary) FROM dmdl.employee
HAVING SUM(salary) > 15000
GROUP BY department_id
ORDER BY SUM(salary) DESC;

-- 过滤查询
-- 比较运算符
```

= 等于
> 大于
>= 大于等于
< 小于
<= 小于等于
<> 不等于

-- in 与列表中的值进行匹配
-- 查询 id 为 101 和 103 的名字
SELECT employee_name FROM dmdl.employee
WHERE employee_id IN(101,103);

-- like 模糊查询
like_ 匹配一个字符
like% 匹配多个字符
-- 查询以“王”字开头的名字
SELECT employee_name FROM dmdl.employee
LIKE '王%';
-- 查询以“明”字结尾的名字
SELECT employee_name FROM dmdl.employee
LIKE '%明';
-- 查询包含“漫”字的名字
SELECT employee_name FROM dmdl.employee
LIKE '%漫%';
-- 查询第二个字为静的名字
SELECT employee_name FROM dmdl.employee
LIKE '_静%';

-- 非空查询 is not null
SELECT employee_id FROM dmdl.employee IS NOT NULL;
-- 查询空值
SELECT employee_id FROM dmdl.employee IS NULL;

-- 逻辑运算符
-- and 满足所有条件，则返回数据
-- or 满足一个条件即返回数据
-- not 条件为假时则返回数据

-- 查询工资大于 6000 和部门为 2 的员工名字
SELECT employee_name FROM dmdl.employee
WHERE salary > 6000
AND department_id = 2;
-- 查询工资大于 6000 或部门为 2 的员工
SELECT employee_name FROM dmdl.employee

```
WHERE salary > 6000
OR department_id = 2;

-- 连接查询内连接
-- 交叉查询（避免）CROSS JOIN
SELECT a.NAME,b.NAME
FROM dmdl.table1 a CROSS JOIN dmdl.table2 b;
-- 等价于
SELECT a.NAME,b.NAME
FROM dmdl.table1 a,dmdl.table2 b;

-- 自然连接查询 NATURAL JOIN,表，需要具有相同的列名并且相同的数据类型
select a.NAME,b.NAME
FROM dmhr.table1 a NATURAL JOIN dmhr.table2 b;

-- using 表连接条件
select a.NAME,b.NAME
FROM dmhr.table1 a JOIN dmhr.table2 b
using (department_id);

-- on 查询（重点）
SELECT a.NAME,b.NAME
FROM dmhr.table1 a   JOIN dmhr.table2 b
ON(table1.column_name = table2.column_name);
-- 等价于
select a.NAME,b.NAME
FROM dmhr.table1 a ,dmhr.table2 b
where a.DEPARTMENT_ID=b.DEPARTMENT_ID;


-- 连接查询外连接
-- 左外连接查询
--返回左边表的全部记录，右边表返回满足条件的记录，不满足的就用 null 代替。
select a.ID,b.ADDR
from dmhr.TEST1 a left OUTER join dmhr.test2
bon a.NAME=b.NAME;
-- 右外连接查询
返回右边表的全部记录，左边边表返回满足条件的记录，不满足的就用 null 代替。
select a.ID,b.ADDR
from dmhr.TEST1 a right outer join dmhr.test2
bon a.NAME=b.NAME;
-- 全外连接查询 -- 左外连接+右外连接
select a.ID,b.ADDR
from dmhr.TEST1 a full outer join dmhr.test2
```

bon a.NAME=b.NAME;

-- 表空间管理
-- 查询表空间
SELECT * FROM dba_tablespaces;

-- 创 建 表 空 间 TEST(使用图形化)
--数据文件/home/dmdba/dmdbms/data/DAMENG/TEST01.DBF,
--初始大小为 50M,
--开启自动扩展，每次扩展 1M，最大可扩展至 1G.
create tablespace "TEST"
datafile'/home/dmdba/dmdbms/data/DAMENG/TEST01.DBF'
size 50
autoextendonnext 1 maxsize 1024 CACHE = NORMAL;

--查看当前用户
SELECT user;
--查看当前用户内容
SELECT dba_users;
--USERNAME:用户名
--ACCOUNT_STATUS：用户的状态，open 状态才能连接数据库
--LOCK_DATE：用户的锁定时间
--EXPIRY_DATE：多长时间过期，过期之后，下次登录数据库时需要更改密码
--DEFAULT_TABLESPACE：用户的默认表空间

-- DM 用户
分为预定义用户和自定义用户。
SYS：系统用户，不能登录，存放数据库字典的信息。
SYSDBA：数据库管理员
SYSAUDITOR：数据库审计员，创建审计规则，查看审计记录
SYSSSO：数据库安全员，创建安全规则

-- 权限
CREATE table; 创建表
ALTER database; 修改数据库
CREATE database; 创建数据库
alert tablespace; 修改表空间
DROP tablespace; 删除表空间
CREATE user; 创建用户
CREATE view; 创建视图
CREATE procedure; 创建存储过程
CREATE role; 创建角色
CREATE schema; 创建模式
CREATE index; 创建索引

```
-- 创建用户 DMTEST，（图形化）
--密码是 Dameng123,
--该用户在登录失败 5 次后锁定 3 分钟,
--密码在 180 天之后自动过期。
--拥有创建表，创建视图，创建索引的权限,
create user "DMTEST"
identified by "Dameng123"
limit FAILED_LOGIN_ATTEMPS 5
PASSWORD_LIFE_TIME 180
PASSWORD_LOCK_TIME 3;
grant CREATE TABLE,CREATE VIEW,CREATE INDEX to "DMTEST";

--角色名一定要大写
--创建角色 ROLE1,
--拥有创建表和创建视图的权限,
--可以查看 dmhr.department
--可以查看 dmhr.employee.hire_date
--可以查看 dmhr.employee.employee_name
--可以查看和修改 dmhr.employee.email
create role "ROLE1";
grant CREATE TABLE,CREATE VIEW to "ROLE1";
grant SELECT on "DMHR"."DEPARTMENT" to "ROLE1";
grant SELECT("HIRE_DATE") on "DMHR"."EMPLOYEE" to "ROLE1";
grant SELECT("EMPLOYEE_NAME") on "DMHR"."EMPLOYEE" to "ROLE1";
grant SELECT("EMAIL"),UPDATE("EMAIL") on "DMHR"."EMPLOYEE" to"ROLE1";

---修改用户 TEST 的密码
alter user "TEST" identified by "dameng123";

---锁定/解锁用户 TEST
alter USER TEST ACCOUNT LOCK;
alter USER TEST ACCOUNT UNLOCK;

---修改用户 TEST 的默认表空间为 sss
alter user "TEST" default tablespace "SSS";

---将 ROLE1 角色授予给 TEST 用户
grant "ROLE1" to "TEST";

---删除用户 TEST
drop user "TEST";

-- 想要访问其他模式下面的表，不想添加模式名？
```

```
--切换模式为 DMHR
set SCHEMA DMHR;

-- 表管理（图形化）
-- 对表的数据做完 insert,update,delete 后一定要记得提交，
--不提交等于白做，提交 commit。

--创建表，非空约束 id 列
create table 'test'.'TEST1' (id int not null);

-- 唯一约束 name 列（图形化）
唯一约束要求数据唯一，创建唯一约束会自动创建唯一索引
CREATE TABLE 'TEST'.'TEST2'
("ID" INT, "NAME" VARCHAR(50), UNIQUE("NAME"))
STORAGE(ON "MAIN", CLUSTERBTR) ;

--主键约束（必考）（图形化）
--主键约束就是非空+唯一，一个表只能有一个主键
CREATE TABLE 'TEST'.'TEST3'
("ID" INTEGER NOT NULL, "NAME" VARCHAR(50),
NOT CLUSTER PRIMARY KEY("ID"))
STORAGE(ON "TEST", CLUSTERBTR) ;

--外键约束（必考）
--外键约束也叫参照约束或者完整性约束
CREATE TABLE 'TEST'.'TEST4'
("ID" INTEGER, "NAME" CHAR(10) NOT NULL,
NOT CLUSTER PRIMARY KEY("NAME"), FOREIGN KEY("ID")
REFERENCES "TEST"."TEST3"("ID"))
STORAGE(ON"TEST", CLUSTERBTR) ;

-- 有主外键约束关系的表，
--在插入数据时，要先插入主键约束所在表的数据，
--再插入外键约束所在表的数据；
--在删除数据时，
--要先删除外键约束所在表的数据，
--再删除主键约束所在表的数据。


-- 检查约束
CREATE TABLE "TEST"."TEST5"
("ID" INT, CHECK(ID > 8))
STORAGE(ON "TEST", CLUSTERBTR) ;
```

```
-- 更新数据
update test.test2 set name='FFF' where id=4;

-- 修改表（图形化）
--增加列
alter table "TEST"."TEST1" add column("ADDR" VARCHAR(100));
--修改列
alter table "TEST"."TEST1" modify "ADDR" VARCHAR(50);
--对列重命名
alter table test.test1 rename COLUMN addr to addr1;
--对表重命名
alter table test."test1" rename to TEST1;
--删除列
alter table "TEST"."TEST1" drop column "ADDR";

--删除表:
drop table "TEST"."TEST1" restrict;


--视图
-- 创建视图
Create view 模式名.视图名
as select .....;
--删除视图:
drop view "模式名"."视图名" restrict;


-- 索引
--创建索引需要注意：索引名大写，索引列是否正确。
create index "模式名"."视图名"
ON "模式名"."视图名"("要创建索引的列") ;

-- 创建文件夹 backup
mkdir -p /home/dmdba/dmdbms/BACKUP

-- 备份数据库
--冷备（图形化）
./console

-- 热备
--创建归档路径（图形化）
mkdir -p /home/dmdba/dmdbms/arch
```