

01.MySQL数据库基础

第一章 数据库基础

一、单元概述

通过本章的学习能够了解MySQL语言的基础概念，MySQL数据库的软件安装过程以及MySQL数据库可视化工具的操作

二、教学重点与难点

重点：

- MySQL软件安装

难点：

- 构建MySQL实验数据的环境

1.1 数据库基础概述

1.1.1 数据库管理系统概述

- 数据库（DB）是一种专门存储信息和维护信息的容器，严格地说数据库是“按照数据结构来组织、存储和管理信息的仓库”。

- 数据库管理系统（Database Management System - DBMS）管理数据库的软件。具有对数据存储、安全、一致性、并发操作、恢复和访问等功能。

1.1.2 数据库特征

- 数据结构化
- 实现数据共享
- 减少数据冗余
- 数据独立性

1.1.3 数据库类型

- 网状型数据库
- 层次型数据库
- 关系型数据库

1.1.4 关系型数据库

- 关系型数据库管理系统（RDBMS）是应用最广泛的一种数据库管理系统，关系型数据库管理系统以表、字段和记录等结构来组织数据。表用来保存数据，每个表由一组字段来定义其结构，记录则是表中的一条数据。

部门号	名称	工号	姓名	年龄	部门号
1001	研发部	1	小牛马	20	1001
1002	销售部	2	大头	20	1001
1003	人力部	3	小鹏	20	1001
		4	Nezuko	20	1002
		5	Whisper	3	1002
		6	几何心凉	20	1003
		7	橘猫	1	1003

部门表

员工表

- 关系数据库是指一些相关的表和其他数据库对象的集合。
对于关系数据库来说，关系就是表的同义词。
- 表是由行和列组成（类似二维数组的结构）。
 - 列包含一组命名的属性（也称字段）。
 - 行包含一组记录，每行包含一条记录。
 - 行和列的交集称为数据项，指出了某列对应的属性在某行上的值，也称为字段值。
 - 列需定义数据类型，比如整数或者字符型的数据。
- 关系数据库的数据结构图示：

列，字段，属性

LAST_NAME	HIRE_DATE	SALARY
Zlotkey	2000-1-29	10500.00
Tucker	1997-1-30	10000.00
Bernstein	1997-3-24	9500.00
Hall	1997-8-20	9000.00
Olsen	1998-3-30	8000.00
Cambrault	1998-12-9	7500.00
Tuvault	1999-11-23	7000.00
King	1996-1-30	10000.00
Sully	1996-3-4	9500.00
McEwen	1996-8-1	9000.00

行，记录，元组

表/关系

数据单元、数据项、属性值、字段值

1.1.5 常见关系数据库

- Oracle
- DB2
- Sybase
- Microsoft SQL Server
- MySQL

MySQL就是当前Web开发中尤其是JavaWEB开发中使用最为广泛的数据库。

1.2 MySQL数据库

1.2.1 MySQL数据库系统简介

- MySQL是由瑞典 MySQL AB公司开的一种开放源代码的关系型数据库管理系统（RDBMS），目前属于 Oracle 旗下产品。MySQL数据库系统使用最常用的数据库管理语言——结构化查询语言（SQL）进行数据库管理。由于MySQL是开放源代码的，因此任何人都可以在General Public License的许可下下载并根据个性化的需要对其进行修改。MySQL因为其速度、可靠性和适应性而备受关注。大多数人都认为在不需要事务化处理的情况下，MySQL是管理内容最好的选择。

1.2.2 MySQL 8 安装

1 解压到 C:\mysql-8.0.18-winx64

2 配置 my.ini

实测发现没有my.ini文件，在安装根目录（C:\mysql-8.0.18-winx64）下创建 my.ini

3 配置my.ini 其内容如下

```
[mysqld]
```

```
# 设置3306端口
```

```
port=3306
```

```
# 设置mysql的安装目录
```

```
basedir=C:\\mysql-8.0.18-winx64    # 切记此处一定要用双斜杠\\，单斜杠我这里会出错，不过看别人的教程，有的是单斜杠。自己尝试吧
```

```
# 设置mysql数据库的数据的存放目录
datadir=C:\\mysql-8.0.18-winx64\\data # 此处
同上
```

```
# 允许最大连接数
max_connections=200
```

```
# 允许连接失败的次数。这是为了防止有人从该主机试图攻击
数据库系统
max_connect_errors=10
```

```
# 服务端使用的字符集默认为UTF8
character-set-server=utf8
```

```
# 创建新表时将使用的默认存储引擎
default-storage-engine=INNODB
```

```
# 默认使用“mysql_native_password”插件认证
default_authentication_plugin=mysql_native_pa
ssword
```

```
[mysql]
# 设置mysql客户端默认字符集
default-character-set=utf8
```

```
[client]
# 设置mysql客户端连接服务端时默认使用的端口
```

port=3306

default-character-set=utf8

注意 其中的data目录不需要创建，下一步初始化工作中会自动创建。

4 安装mysql

以管理员身份运行cmd 在MySQL安装目录的 bin 目录下执行命令：

```
>mysqld --initialize --console
```

执行完成后，会打印 root 用户的初始默认密码，比如：

```
>mysqld --initialize --console
```

```
2018-04-28T15:57:17.087519Z 0 [System] [MY-013169] [Server] C:\Program
```

```
Files\MySQL\bin\mysqld.exe (mysqld 8.0.11)
```

```
initializing of server in progress as process 4984
```

```
2018-04-28T15:57:24.859249Z 5 [Note] [MY-010454] [Server] A temporary password is
```

```
generated for root@localhost: rI5rvf5x5G,E
```

```
2018-04-28T15:57:27.106660Z 0 [System] [MY-013170] [Server] C:\Program
```

```
Files\MySQL\bin\mysqld.exe (mysqld 8.0.11)
```

```
initializing of server has completed
```

#注意! 要是你手贱, 关快了, 或者没记住, 那也没事, 删掉初始化的 `datadir` 目录, 再执行一遍初始化命令, 又会重新生成的。

5 安装服务

在MySQL安装目录的 `bin` 目录下执行命令:

```
mysqld --install [服务名] ==比如 mysqld --install mysql8 :Yjv.Gqf=2EN
```

后面的服务名可以不写, 默认的名字为 `mysql`, 也可自行指定

安装完成之后, 就可以通过命令 `net start mysql8` 启动MySQL的服务了。通过命令 `net stop mysql8` 停止服务。

通过命令 `sc delete MySQL/mysqld -remove` 卸载MySQL 服务

6 更改密码

在MySQL安装目录的 `bin` 目录下执行命令: `mysql -u root -p`, 输入刚才记住的密码, 然后执行

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'root'; #后面的root是指的新密码
```

修改密码, 注意命令尾的; 一定要有, 这是mysql的语法

5 安装服务

在MySQL安装目录的 bin 目录下执行命令：

```
mysqld --install [服务名] ==比如 mysqld --install mysql8 :Yjv.Gqf=2EN
```

后面的服务名可以不写，默认的名字为 mysql，也可自行指定

安装完成之后，就可以通过命令net start mysql8启动MySQL的服务了。通过命令net stop mysql8停止服务。

通过命令sc delete MySQL/mysqld -remove卸载MySQL 服务

6 更改密码

在MySQL安装目录的 bin 目录下执行命令：mysql -u root -p，输入刚才记住的密码，然后执行

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'root'; #后面的root是指的新密码
```

修改密码，注意命令尾的；一定要有，这是mysql的语法

7 可以用 命令查看一下默认安装的数据库：

```
show databases;
```

```
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| newbee_mall_db |
```

```
| performance_schema |
| sys                |
+-----+
```

```
use mysql;
```

```
show tables;
```

```
+-----+
| Tables_in_mysql |
+-----+
| columns_priv    |
| component       |
| db              |
| default_roles   |
| engine_cost     |
| func            |
| general_log     |
| global_grants   |
| gtid_executed   |
| help_category   |
| help_keyword    |
| help_relation   |
| help_topic      |
| innodb_index_stats |
| innodb_table_stats |
| password_history |
| plugin          |
| procs_priv      |
| proxies_priv    |
| role_edges      |
```

server_cost	
servers	
slave_master_info	
slave_relay_log_info	
slave_worker_info	
slow_log	
tables_priv	
time_zone	
time_zone_leap_second	
time_zone_name	
time_zone_transition	
time_zone_transition_type	
user	
+-----+	

其中user表里面存储MySQL用户信息。我们可以看一下默认MySQL用户：

```
mysql> select user,host,authentication_string
from user;
```

+-----+	+-----+	+-----+

-----+		
user	host	
authentication_string		
+-----+	+-----+	+-----+

-----+		

```

| mysql.infoschema | localhost |
$A$005$THISISACOMBINATIONOFINVALIDSALTANDPASS
WORDTHATMUSTNEVERBRBEUSED |
| mysql.session    | localhost |
$A$005$THISISACOMBINATIONOFINVALIDSALTANDPASS
WORDTHATMUSTNEVERBRBEUSED |
| mysql.sys        | localhost |
$A$005$THISISACOMBINATIONOFINVALIDSALTANDPASS
WORDTHATMUSTNEVERBRBEUSED |
| root              | localhost |
*81F5E21E35407D884A6CD4A731AEBFB6AF209E1B
|
+-----+-----+-----+
-----
-----+

```

管理员root的host是localhost，代表仅限localhost登录访问。如果要允许开放其他ip登录，则需要添加新的host。如果要允许所有ip访问，可以直接修改成“%”

创建用户：

```
CREATE USER 'xxh'@'%' IDENTIFIED WITH
mysql_native_password BY 'xxh123!@#';
```

#(需要注意：mysql8.0加密方式修改了)

#检查用户

```
select user, host, plugin,  
authentication_string from user\G; //这个 \G  
参数只是改变显示格式的,加了它以后不显示表格线,没什么大  
用
```

#授权远程数据库

#授权所有权限

```
GRANT ALL PRIVILEGES ON *.* TO 'xxh'@'%';
```

#授权基本的查询修改权限, 按需求设置

GRANT

```
SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER  
ON *.* TO 'xxh'@'%';
```

查看用户权限

```
show grants for 'xxh'@'%';
```

1.2.3 启动服务

- 打开运行窗口windows+r
- 进入服务列表services.msc
- 找到Mysql服务名称
- 鼠标右键开启服务

1.2.4 停止服务

- 打开运行窗口windows+r
- 进入服务列表services.msc
- 找到Mysql服务名称

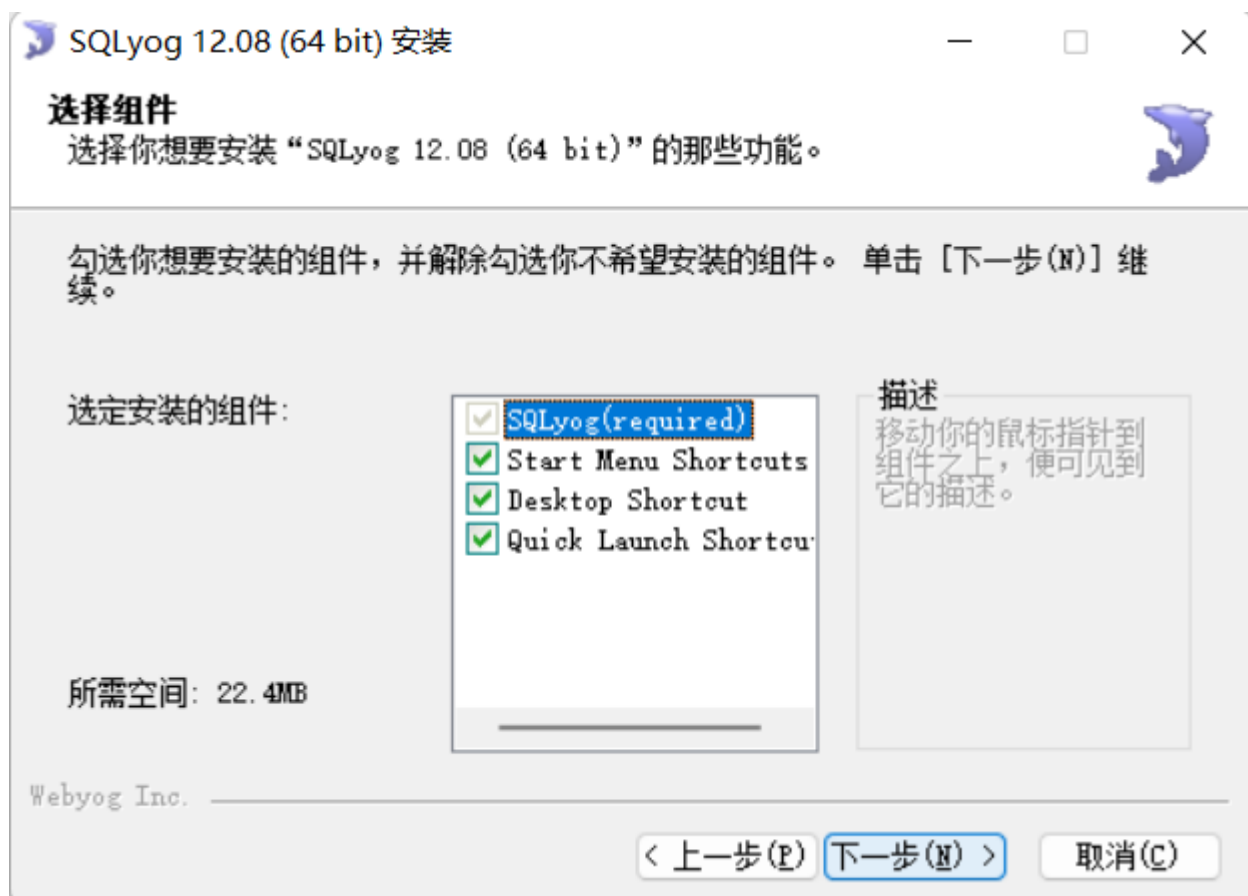
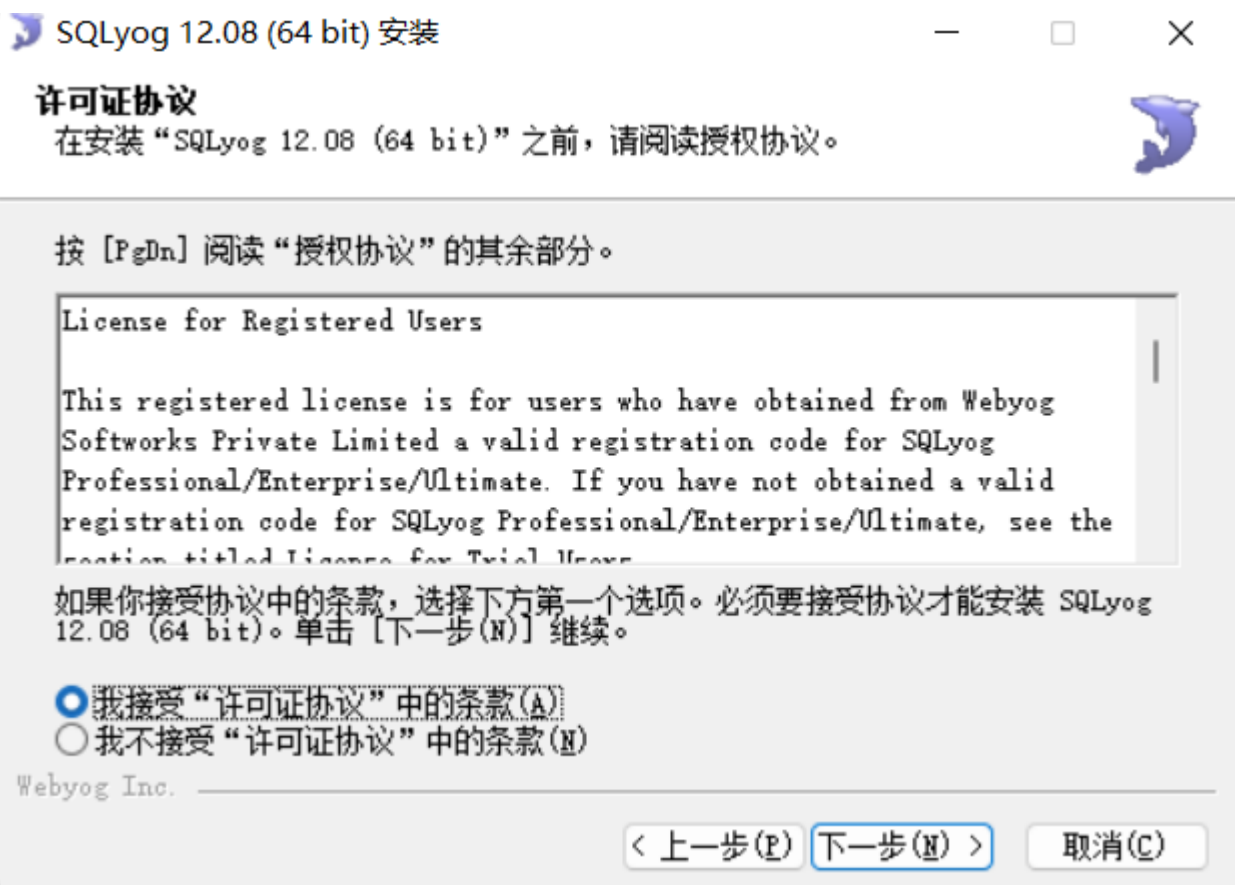
- 鼠标右键停止服务

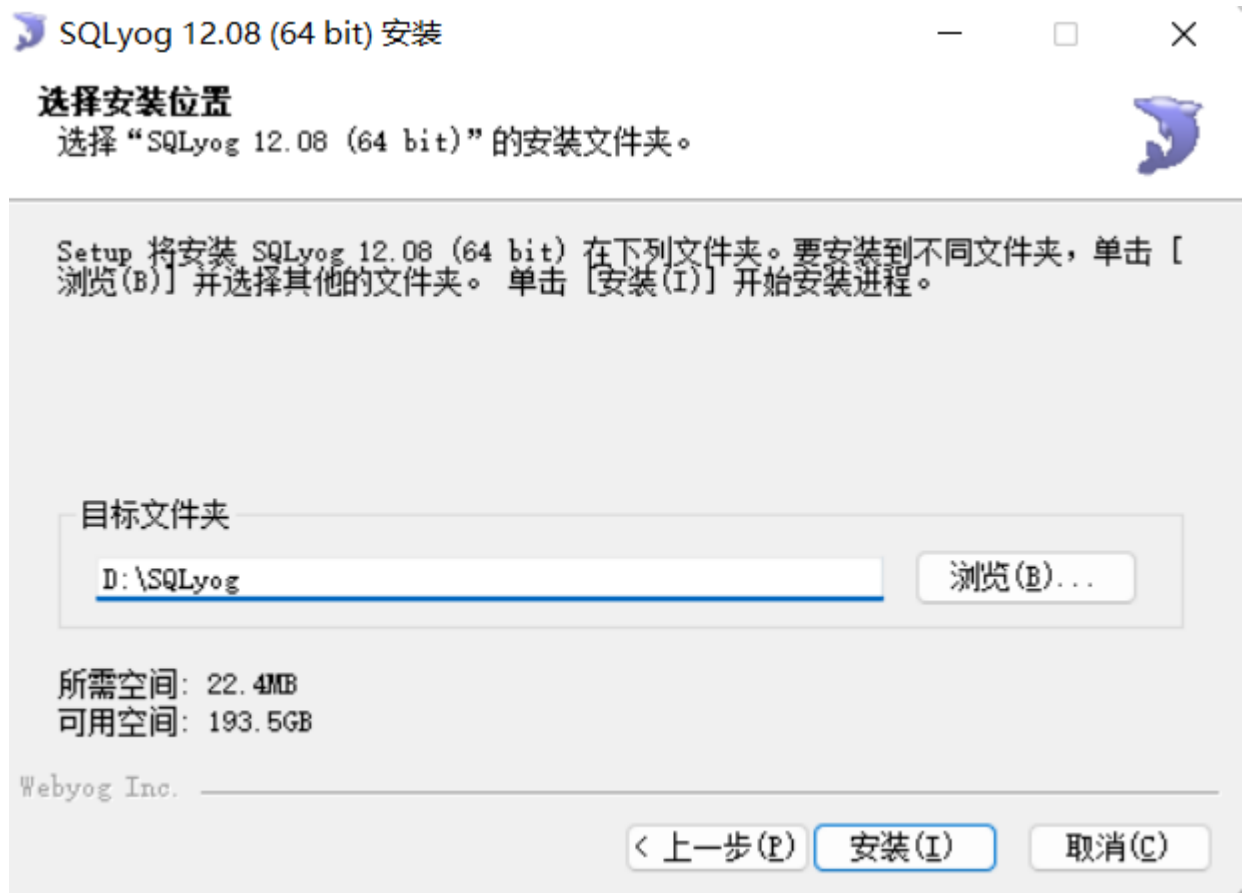
1.3 MySQL可视化工具

1.3.1 安装SQLyog可视化开发工具

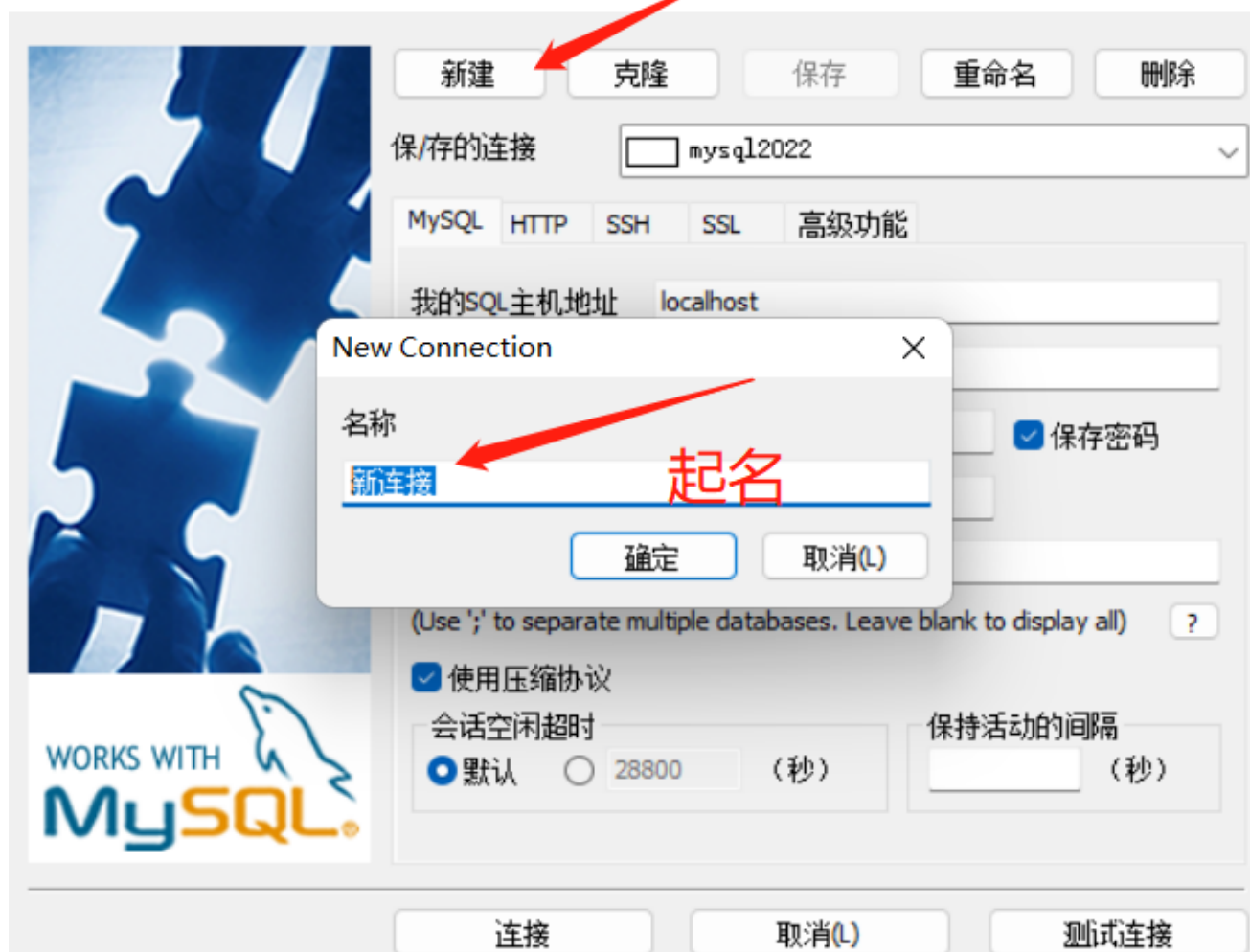
- 下载安装包：<https://sqlyog.en.softonic.com/download>
- 解压出Webyog_SQLyog_setup.exe安装程序双击运行开始安装。







1.3.2 SQLyog可视化开发工具使用



新建

克隆

保存

重命名

删除

保/存的连接

mysql2022

MySQL

HTTP

SSH

SSL

高级功能

我的SQL主机地址

localhost

用户名

root

密码

••••

输入密码

☒ 保存密码

端口

3306

数据/库

(Use ';' to separate multiple databases. Leave blank to display all)

?

☒ 使用压缩协议

会话空闲超时

☒ 默认
☐ 28800

(秒)

保持活动的间隔

(秒)

连接

取消(L)

测试连接

点击测试连接



WORKS WITH
MySQL


新建 克隆 保存 重命名 删除


保/存的连接

MySQL HTTP SSH SSL 高级功能

我的SQL主机地址

连接信息

 Connection successful!
MySQL version : 8.0.18



确定

会话空闲超时
☒ 默认 ☐ 28800 (秒)

保持活动的间隔

(秒)

连接 取消(L) 测试连接

连接到我的SQL主机

新建克隆保存重命名删除

保/存的连接mysql2022

MySQLHTTPSSHSSL高级功能

我的SQL主机地址localhost

用户名root

密码

保存密码

端口3306

数据/库

(Use ';' to separate multiple databases. Leave blank to display all)

☒ 使用压缩协议

会话空闲超时

默认28800(秒)

保持活动的间隔

连接

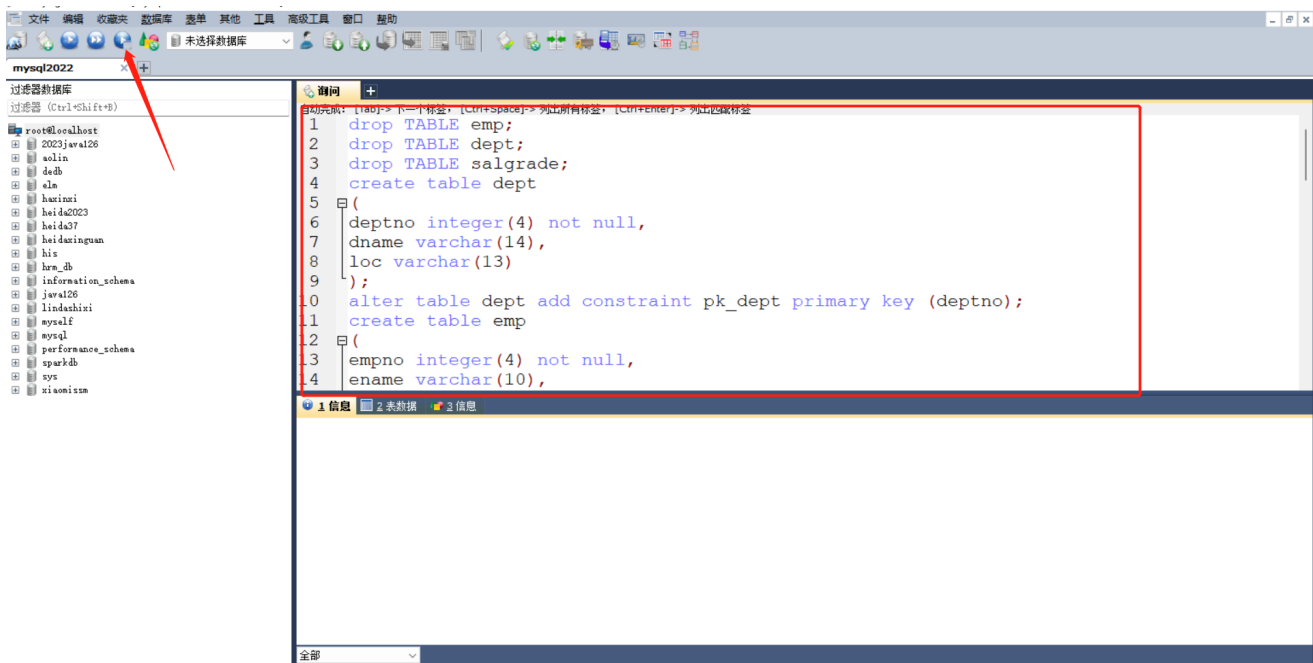
取消(L)

测试连接

The screenshot shows the SQLyog Ultimate 64 interface. On the left, there is a 'Database List' (数据库列表区域) showing a list of databases including 'root@localhost', '2023java126', 'eolun', 'de-db', 'els', 'hazimoi', 'heid2023', 'heid207', 'heid207', 'heid207', 'heid207', 'his', 'hsa_db', 'information_schema', 'java126', 'lindashisi', 'mysql', 'mysql', 'performance_schema', 'sparkdb', 'sys', and 'xiao123456'. On the right, there is a 'Query Editor' (编写代码区域) with a text area for writing SQL queries. The interface also includes a menu bar, a toolbar, and a status bar at the bottom.

1.4 构建测试数据

- 将以下脚本SQL语句全部复制到test库下的查询窗口并执行



```
drop TABLE emp;  
drop TABLE dept;  
drop TABLE salgrade;  
create table dept  
(  
deptno integer(4) not null,  
dname varchar(14),  
loc varchar(13)  
);  
alter table dept add constraint pk_dept  
primary key (deptno);  
create table emp  
(  
empno integer(4) not null,  
ename varchar(10),  
job varchar(9),
```

```
mgr integer(4),
hiredate date,
sal decimal(7,2),
comm decimal(7,2),
deptno integer(2)
);
alter table emp add constraint pk_emp primary
key (empno);
alter table emp add constraint fk_deptno
foreign key (deptno) references dept
(deptno);
create table salgrade
(
grade integer(1),
losal decimal(7,2),
hisal decimal(7,2)
);
insert into DEPT (DEPTNO, DNAME, LOC) values
(10, 'ACCOUNTING', 'NEW YORK');
insert into DEPT (DEPTNO, DNAME, LOC) values
(20, 'RESEARCH', 'DALLAS');
insert into DEPT (DEPTNO, DNAME, LOC) values
(30, 'SALES', 'CHICAGO');
insert into DEPT (DEPTNO, DNAME, LOC) values
(40, 'OPERATIONS', 'BOSTON');
insert into SALGRADE (GRADE, LOSAL, HISAL)
values (1, 700, 1200);
```

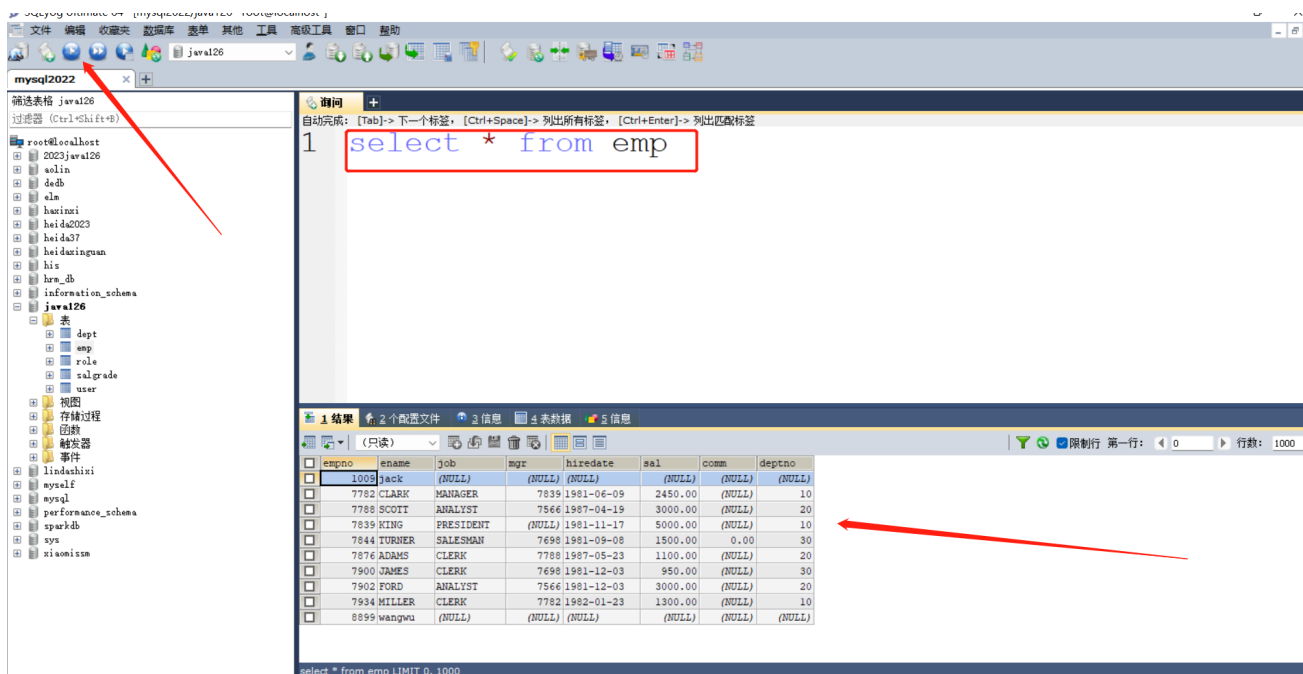
```
insert into SALGRADE (GRADE, LOSAL, HISAL)
values (2, 1201, 1400);
insert into SALGRADE (GRADE, LOSAL, HISAL)
values (3, 1401, 2000);
insert into SALGRADE (GRADE, LOSAL, HISAL)
values (4, 2001, 3000);
insert into SALGRADE (GRADE, LOSAL, HISAL)
values (5, 3001, 9999);
-- 下面要特别注意, %d-%m-%Y 里的 d 和 m 一定要小写
insert into EMP (EMPNO, ENAME, JOB, MGR,
HIREDATE, SAL, COMM, DEPTNO)
values (7369, 'SMITH', 'CLERK', 7902,
STR_TO_DATE('17-12-1980', '%d-%m-%Y'),
800.00, null, 20);
insert into EMP (EMPNO, ENAME, JOB, MGR,
HIREDATE, SAL, COMM, DEPTNO)
values (7499, 'ALLEN', 'SALESMAN', 7698,
STR_TO_DATE('20-02-1981', '%d-%m-%Y'),
1600.00, 300.00, 30);
insert into EMP (EMPNO, ENAME, JOB, MGR,
HIREDATE, SAL, COMM, DEPTNO)
values (7521, 'WARD', 'SALESMAN', 7698,
STR_TO_DATE('22-02-1981', '%d-%m-%Y'),
1250.00, 500.00, 30);
insert into EMP (EMPNO, ENAME, JOB, MGR,
HIREDATE, SAL, COMM, DEPTNO)
```

```
values (7566, 'JONES', 'MANAGER', 7839,  
STR_TO_DATE('02-04-1981', '%d-%m-%Y'),  
2975.00, null, 20);  
insert into EMP (EMPNO, ENAME, JOB, MGR,  
HIREDATE, SAL, COMM, DEPTNO)  
values (7654, 'MARTIN', 'SALESMAN', 7698,  
STR_TO_DATE('28-09-1981', '%d-%m-%Y'),  
1250.00, 1400.00, 30);  
insert into EMP (EMPNO, ENAME, JOB, MGR,  
HIREDATE, SAL, COMM, DEPTNO)  
values (7698, 'BLAKE', 'MANAGER', 7839,  
STR_TO_DATE('01-05-1981', '%d-%m-%Y'),  
2850.00, null, 30);  
insert into EMP (EMPNO, ENAME, JOB, MGR,  
HIREDATE, SAL, COMM, DEPTNO)  
values (7782, 'CLARK', 'MANAGER', 7839,  
STR_TO_DATE('09-06-1981', '%d-%m-%Y'),  
2450.00, null, 10);  
insert into EMP (EMPNO, ENAME, JOB, MGR,  
HIREDATE, SAL, COMM, DEPTNO)  
values (7788, 'SCOTT', 'ANALYST', 7566,  
STR_TO_DATE('19-04-1987', '%d-%m-%Y'),  
3000.00, null, 20);  
insert into EMP (EMPNO, ENAME, JOB, MGR,  
HIREDATE, SAL, COMM, DEPTNO)  
values (7839, 'KING', 'PRESIDENT', null,  
STR_TO_DATE('17-11-1981', '%d-%m-%Y'),  
5000.00, null, 10);
```



```
insert into EMP (EMPNO, ENAME, JOB, MGR,  
HIREDATE, SAL, COMM, DEPTNO)  
values (7844, 'TURNER', 'SALESMAN', 7698,  
STR_TO_DATE('08-09-1981', '%d-%m-%Y'),  
1500.00, 0.00, 30);  
insert into EMP (EMPNO, ENAME, JOB, MGR,  
HIREDATE, SAL, COMM, DEPTNO)  
values (7876, 'ADAMS', 'CLERK', 7788,  
STR_TO_DATE('23-05-1987', '%d-%m-%Y'),  
1100.00, null, 20);  
insert into EMP (EMPNO, ENAME, JOB, MGR,  
HIREDATE, SAL, COMM, DEPTNO)  
values (7900, 'JAMES', 'CLERK', 7698,  
STR_TO_DATE('03-12-1981', '%d-%m-%Y'),  
950.00, null, 30);  
insert into EMP (EMPNO, ENAME, JOB, MGR,  
HIREDATE, SAL, COMM, DEPTNO)  
values (7902, 'FORD', 'ANALYST', 7566,  
STR_TO_DATE('03-12-1981', '%d-%m-%Y'),  
3000.00, null, 20);  
insert into EMP (EMPNO, ENAME, JOB, MGR,  
HIREDATE, SAL, COMM, DEPTNO)  
values (7934, 'MILLER', 'CLERK', 7782,  
STR_TO_DATE('23-01-1982', '%d-%m-%Y'),  
1300.00, null, 10);
```

- 查看测试数据



1.5 存储引擎

- 存储引擎就是如何存储数据、如何为存储的数据建立索引和如何更新、查询数据等技术的实现方法。因为在关系数据库中数据的存储是以表的形式存储的，所以存储引擎简而言之就是指表的类型。数据库的存储引擎决定了表在计算机中的存储方式。
- 在Oracle和SQL Server等数据库中只有一种存储引擎，所有数据存储管理机制都是一样的。而MySQL数据库提供了多种存储引擎，用户可以根据不同的需求为数据表选择不同的存储引擎，用户也可以根据自己的需要编写自己的存储引擎，MySQL的核心就是存储引擎。

1.5.1 InnoDB存储引擎

- InnoDB存储引擎的特点:

- ○ 支持外键（Foreign Key）
- ○ 支持事务（Transaction）：如果某张表主要提供OLTP支持，需要执行大量的增、删、改操作（insert、delete、update语句），出于事务安全方面的考虑，InnoDB存储引擎是更好的选择。
- ○ 最新版本的MySQL已经开始支持全文检索。

1.5.2 MyISAM存储引擎

- MyISAM存储引擎的特点：
- ○ MyISAM具有检查和修复表的大多数工具。
- ○ MyISAM表可以被压缩
- ○ MyISAM表最早支持全文索引
- ○ 但MyISAM表不支持事务
- ○ 但MyISAM表不支持外键（Foreign Key）。
- ○ 如果需要执行大量的select语句，出于性能方面的考虑，MyISAM存储引擎是更好的选择。

1.5.3 MEMORY存储引擎

- MEMORY存储引擎的特点：
- ○ MEMORY存储引擎是MySQL中一类特殊的存储引擎。该存储引擎使用存在于内存中的内容来创建表，每个表实际对应一个磁盘文件，格式为.frm。这类表因为数据在内存中，且默认使用HASH索引，所以访问速度非常快；但一旦服务关闭，表中的数据会丢失。

- 每个MEMORY表可以放置数据量的大小受max_heap_table_size系统变量的约束，初始值为16MB，可按需求增大。此外，在定义MEMORY表时可通过MAX_ROWS子句定义表的最大行数。
- 该存储引擎主要用于那些内容稳定的表，或者作为统计操作的中间表。对于该类表需要注意的是，因为数据并没有实际写入磁盘，一旦重启，则会丢失。
- MySQL5.7默认的存储引擎是InnoDB。

1.5.4 存储引擎的选择

- 不同存储引擎都有各自的特点，以适应不同的需求
- MySQL存储引擎功能对比

功 能	InnoDB	MyISAM	Memory
存储限制	64TB	256TB	RAM
支持事务	支持	无	无
空间使用	高	低	低
内存使用	高	低	高
支持数据缓存	支持	无	无
插入数据速度	低	高	高
支持外键	支持	无	无

1.6 MySQL字符集

- 字符(Character)是指人类语言最小的表义符号，例如'A'、'B'等。

- 给定一系列字符，对每个字符赋予一个数值，用数值来代表对应的字符，这个数值就是字符的编码(Character Encoding)。
- 给定一系列字符并赋予对应的编码后，所有这些字符和编码对组成的集合就是字符集(Character Set)。MySQL中提供了多种字符集，例如latin1、utf8、gbk等。
- 字符序(Collation)是指在同一字符集内字符之间的比较规则。只有确定字符序后，才能在一个字符集上定义什么是等价的字符，以及字符之间的大小关系。每个字符序唯一对应一种字符集，一个字符集可以对应多种字符序，其中有一个是默认字符序(Default Collation)。
- MySQL中的字符序命名规则：以字符序对应的字符集名称开头，以国家名居中（或以general居中），以ci或cs或bin末尾。例如：latin1字符集对应有latin1_swedish_ci、latin1_spanish_ci、latin1_german1_ci等字符序。以ci结尾的字符序表示大小写不敏感；以cs结尾的字符序表示大小写敏感；以bin结尾的字符序表示按编码值比较。例如在字符序gbk_general_ci规则中，字符'a'和'A'是等价的。
- MySQL提供的下列命令可以在不影响其它数据库字符集的基础上临时修改当前的字符集
 - SET character_set_client = gbk;
 - SET character_set_connection = gbk;
 - SET character_set_database = gbk;

- SET character_set_results = gbk;
- SET character_set_server = gbk;
- SET collation_connection = gbk_chinese_ci ;
- SET collation_database = gbk_chinese_ci ;
- SET collation_server = gbk_chinese_ci ;