数据库入门

一、数据库及其发展史

(1).什么是数据库?

简单的说，数据库(Database)就是一个存放电子计算机数据的仓库，该仓库按照一定的规则和标准结合数据结构和算法对数据进行组织和存储。我们可以通过指定的方式来对数据库中的数据进行管理和应用，从而发挥数据的价值。

(2).数据库的种类

> 按照早期的数据库理论来划分，比较流行数据库模型分为三种，分别是层次模型，网状模型，和关系模型。

在当今的互联网中，最常用的数据库主要分为了**关系型数据库**和**非关系型数据库**.

a.早期数据库

1. 萌芽阶段:所有的的数据都是存储在文件中的, 安全性低, 操作性繁琐.
2. 层次型数据库:

   优点：数据结构比较简单清晰，数据库的查询效率高，提供了良好的完整性支持。 缺点：现实世界中很多联系是非层次性的，它不适用于结点之间具有多对多联系；查询子女结点必须通过双亲结点；由于结构严密，层次命令趋于程序化。
3. 网状型数据库: 没有解决导航问题, 解决了数据完整性问题。

   优点：能够更为直接地描述现实世界，如一个结点可以有多个双亲，结点直接可以有多种联系；具有良好的性能，存取效率较高。 缺点：结构比较复杂，随应用环境的扩大，数据库的结构就变得越来越复杂，不利于最终用户掌握；网状模型的DDL、DML复杂，并且要嵌入某一种高级语言（C、COBOL）中，用户不容易掌握和使用；由于记录之间的联系是通过存取路径实现的，应用程序在访问数据时必须选择适当的存取路径，因此用户必须了解系统结构的细节，加重了编写应用程序的负担。
4. 关系型数据库:

   优点：建立在严格的数学概念的基础上；概念单一，无论实体还是实体之间的联系都是用关系来表示。对数据的检索和更新结构也是关系（也就是我们常说的表）；它的存取路径对用户透明，从而具有更高的独立性、更好的安全保密性，简化了程序员的工作个数据库开发建立的工作。 缺点：存取路径的隐蔽导致查询效率不如格式化数据模型。

2. SQL语言

SQL 全拼为 Structured Query Language, 即 “结构化查询语言”。

SQL语言诞生于1974年，是IBM提出的，但是关系型数据库在1970年已经出现，在没有出现SQL语言之前当时的查询语言依赖于复杂的数学逻辑和符号，查询语言成为关系型数据库发展的一个主要瓶颈，SQL的出现使没有接受过数学和计算机编程正规训练的用户也能简便的使用。

SQL 是一种特殊目的的编程语言，是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统；同时也是数据库脚本文件的扩展名。

3. 关系型数据库

RDBMS 即关系数据库管理系统(Relational Database Management System)，是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。 它有如下特点：

- 数据以表格的形式出现
- 每行为各种记录名称
- 每列为记录名称所对应的数据域
- 许多的行和列组成一张表单
- 若干的表单组成database

数据库	SQL 类型	公司
Oracle	PL/SQL	甲骨文
MySQL	My/SQL	甲骨文
SQL-Server	T-SQL	微软
Access	SQL	微软
SQLite	内嵌型小型数据库	移动前端用的比较多

二、数据库相关术语和概念

- **数据库**: 数据库是一些关联表的集合。
- **数据表**: 表是数据的矩阵。在一个数据库中的表看起来像一个简单的电子表格。
- **列**: 一列(数据元素) 包含了相同类型的数据, 例如邮政编码的数据。
- **行**: 一行（=元组，或记录）是一组相关的数据，例如一条用户订阅的数据。
- **冗余**: 存储两倍数据，冗余降低了性能，但提高了数据的安全性。
- **主键**: 主键是唯一的。一个数据表中只能包含一个主键。你可以使用主键来查询数据。
- **外键**: 外键用于关联两个表。
- **复合键**: 复合键（组合键）将多个列作为一个索引键，一般用于复合索引。
- **索引**: 使用索引可快速访问数据库表中的特定信息。索引是对数据库表中一列或多列的值进行排序的一种结构。类似于书籍的目录。
- **参照完整性**: 参照的完整性要求关系中不允许引用不存在的实体。与实体完整性是关系模型必须满足的完整性约束条件，目的是保证数据的一致性。

```
+----+------------+-----+------------+
| id | name       | sex | birthday   |
+----+------------+-----+------------+
|  1 | 郭德纲     | 男  | 1997-10-01 |
|  2 | 陈乔恩     | 女  | 1995-03-02 |
|  3 | 赵丽颖     | 女  | 1995-04-04 |
|  4 | 王宝强     | 男  | 1998-10-05 |
|  5 | 周冬雨     | 女  | 1996-07-09 |
|  6 | 张学友     | 男  | 1993-05-02 |
|  7 | 陈意涵     | 女  | 1994-08-30 |
|  8 | 赵本山     | 男  | 1995-06-01 |
|  9 | 吴宣仪     | 女  | 1997-02-28 |
+----+------------+-----+------------+
```

三、Linux数据库的开启和连接

1. 安装数据库

基于Debian平台的Linux系统，可以直接使用apt命令安装mysql

```
sudo apt install -y mysql-server mysql-client
```

由于历史原因，如果在CentOS里运行`yum install mysql`不会安装mysql数据库，而是会安装MariaDB(关于MySql和MariaDB的关系，请参考百度百科)。在CentOS里安装mysql要稍微复杂一些。

1. 从mysql官网上下载mysql仓库，会得到`mysql80-community-release-el7-3.noarch.rpm`文件。

   ```
   wget https://repo.mysql.com//mysql80-community-release-el7-3.noarch.rpm
   ```

2. 运行`yum install mysql80-community-release-el7-3.noarch.rpm`安装mysql repository.
3. 默认安装的mysql版本是8.0,而我们在开发中常用的是5.7版本，所以需要修改配置文件，默认安装5.7版本。使用`vim vim /etc/yum.repos.d/mysql-community.repo`命令修改`mysql-community.repo`文件。将文件中，mysql80的enable值改为0,mysql57的enable值改为1.

```
[mysql57-community]
name=MySQL 5.7 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.7-community/el/7/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

[mysql80-community]
name=MySQL 8.0 Community Server
baseurl=http://repo.mysql.com/yum/mysql-8.0-community/el/7/$basearch/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
```

4. 运行 `yum install mysql-community-server` 安装mysql数据库。

2. 开启数据库服务

1. Ubuntu : `service mysql start|stop|restart|status`
2. Deepin : `systemctl start|stop|restart|status mysqld`
3. CentOS7 : `systemctl start|stop|restart|status mysqld`
4. CentOS6 : `service mysqld start|stop|restart|status`

3. 连接数据库

各个 Linux 系统连接数据库的语法都一样

语法: `mysql -hloaclhost -uroot -p123456 -P3306`

1. -h : host(ip地址) localhost = 127.0.0.1
2. -u : username(用户账户)
3. -p : password(密码)
4. -P : port(端口, 默认端口3306)

在其他版本的Linux里，root用户的默认密码是空，可以不使用密码直接登录。但是在CentOS7里，mysql安装完成以后，会生成一个临时密码，我们需要通过命令查看到这个默认密码。

```
cat /var/log/mysqld.log |grep password
```

```
[root@iZuf60e5hkfmzk7ol3d801Z ~]# cat /var/log/mysqld.log |grep password
2020-03-15T08:57:54.374518Z 1 [Note] A temporary password is generated for root@localhost: yeTvWMXRF3>9
2020-03-15T08:58:03.963467Z 2 [Note] Access denied for user 'root'@'localhost' (using password: NO)
2020-03-15T09:02:32.639123Z 3 [Note] Access denied for user 'root'@'localhost' (using password: NO)
2020-03-15T09:02:36.864342Z 4 [Note] Access denied for user 'root'@'localhost' (using password: NO)
2020-03-15T09:02:58.457123Z 5 [Note] Access denied for user 'root'@'localhost' (using password: NO)
2020-03-15T09:03:02.636953Z 6 [Note] Access denied for user 'root'@'localhost' (using password: NO)
2020-03-15T09:03:06.467935Z 7 [Note] Access denied for user 'root'@'localhost' (using password: NO)
2020-03-15T09:03:42.427403Z 8 [Note] Access denied for user 'root'@'localhost' (using password: NO)
2020-03-15T09:03:46.950433Z 9 [Note] Access denied for user 'root'@'localhost' (using password: NO)
```

4. 设置root用户密码

数据库连接成功以后，因为此时使用的是root用户的临时密码，此时无法进行任何的操作，需要先修改root用户的密码。

```
alter user root@localhost identified with mysql_native_password by '你的密码';
```

备注 第一次使用 root 连接后最好添加一个新的用户来操作。出于安全考虑，日常开发中最好不要使用 root

```
-- 创建新用户，并设置密码
-- *.* 代表该用户可以操作任何库、任何表
-- 主机名可以使用 '%'，代表允许该用户从任何机器登陆
GRANT ALL PRIVILEGES on *.* to '用户名'@'localhost' IDENTIFIED BY "密码" WITH
GRANT OPTION;

-- 刷新使权限生效
flush privileges;
```

4. 退出数据库

四种方式效果一样:

1. exit
2. quit
3. \q
4. 快捷键: ctrl + d

5. 密码忘记怎么办?

1. 打开配置: `vim /etc/my.cnf`
2. 添加这么一段:

   ```
   [mysqld]
   skip-grant-tables
   ```

   如果文件中已存在 `[mysqld]`, 则直接将 `skip-grant-tables` 写到其下方即可。
3. 修改完成后，保存退出，重启服务: `sudo systemctl restart mysql.service`
4. 使用命令`sudo mysql -uroot`重新连接MySql服务器，此时可以不使用密码直接登录用户。
5. 执行`update mysql.user set authentication_string=password('你的密码') where user="root";`修改root用户的密码。
6. 执行`flush privileges`刷新策略，使策略立刻生效，并退出mysql客户端。
7. 修改`/etc/msyql/mysql.cnf`文件，注释掉第二步添加的两段内容。
8. 运行`sudo systemctl restart mysql.service`重启mysql服务器。
9. 此时可以使用新密码登录mysql服务器。

四、权限管理

1. MySQL 权限的两个阶段

1. 第一阶段为连接验证，主要限制用户连接 mysql-server 时使用的 ip及密码。
2. 第二阶段为操作检查，主要检查用户执行的指令是否被允许，一般非管理员账户不被允许执行 drop、delete 等危险操作。

2. 权限控制安全准则

1. 只授予能满足需要的最小权限，防止用户执行危险操作。

2. 限制用户的登录主机，防止不速之客登录数据库。
3. 禁止或删除没有密码的用户。
4. 禁止用户使用弱密码。
5. 定期清理无效的用户，回收权限或者删除用户。

3. 常用操作

1. 创建账户、权限授予
   - 8.0 之前版本

     ```
     GRANT ALL PRIVILEGES on *.* to '用户名'@'主机' IDENTIFIED BY "密码" WITH
     GRANT OPTION;
     flush privileges; -- 刷新使权限生效
     ```

     - `ALL PRIVILEGES`: 授予全部权限, 也可以指定`select`,`insert`,`update`,`delete`,`create`,`drop`,`index`,`alter`,`grant`,`references`,`reload`,`shutdown`,`process`,`file`十四个权限。
     - `*.*`: 允许操作的数据库和表。
     - 主机名:表示允许用户从哪个主机登录，`%`表示允许从任意主机登录。
   - `WITH GRANT OPTION`: 带有该子句说明允许用户将自己拥有的权限授予别人。
   - 8.0 之后的版本

     ```
     CREATE USER `用户名`@`主机` IDENTIFIED BY '密码';  -- 创建账户
     GRANT ALL ON *.* TO `用户名`@`主机` WITH GRANT OPTION;  -- 授权
     ```

2. 修改密码

   ```
   update user set authentication_string=password('你的密码') where user="root"
   或者
   alter user '用户名'@'主机' identified with mysql_native_password by '你的密码';
   ```

3. 查看权限

   ```
   show grants;  -- 查看当前用户的权限
   show grants for 'abc'@'localhost';  -- 查看用户 abc 的权限
   ```

4. 回收权限

   ```
   revoke all privileges on *.*  from 'abc'@'localhost';  -- 回收用户 abc 的所有
   权限
   revoke grant option on *.* from 'abc'@'localhost';  -- 回收权限的传递
   ```

5. 删除用户

   ```
   use mysql;
   select host, user from user;
   drop user 用户名@'%';
   ```

> 了解:创建一个用户，允许该用户通过主机远程登录。

- 创建一个允许任意主机登录的账号。

```
GRANT ALL PRIVILEGES on *.* to 'zhangsan'@'%' IDENTIFIED BY "密码" WITH GRANT
OPTION;
flush privileges;
```

- 修改mysql配置文件（不同操作系统，不同MySql版本，配置文件存储不同，ubuntu18.04配置文件保存在/etc/mysql/mysql.conf.d/mysqld.cnf; Centos7.7 配合文件保存在 /etc/my.cnf）

```
sudo vim /etc/my.cnf
```

将 `bind-address=127.0.0.1` 代码注释掉（如果这段代码存在的话），让计算机允许mysql远程登录。

- 打开服务器的 3306 端口。
- 使用客户端远程登录到mysql数据库。

```
sudo mysql -uzhangsan -h 远端服务器地址 -P 3306 -p
```

五、数据库的操作

1. 创建数据库

```
create database [if not exists] `数据库名` charset=字符编码(utf8mb4);
```

1. 如果多次创建会报错
2. 如果不指定字符编码，默认为 utf8mb4 (一个汉字占用 4 个字节)
3. 给数据库命名一定要习惯性加上反引号, 防止和关键字冲突

2. 查看数据库

```
show databases;
```

3. 选择数据库

```
use `数据库的名字`;
```

4. 创建数据库

```
create database `数据库名`;
```

5. 修改数据库

```
-- 只能修改字符集
alter database `数据库名` charset=字符集;
```

6. 删除数据库

```
drop database [if exists] `数据库的名字`;
```

六、表的操作

表是建立在数据库中的数据结构，是一类数据的存储集。

1. 表的创建

```
create table [if not exists] `表的名字`(
    id int not null auto_increment primary key comment '主键',
    account char(255) comment '用户名' default 'admin',
    pwd text(16383) comment '密码' not null
)charset=utf8mb4;
```

备注:

- 字符集如果不指定, 默认继承库的字符集.

2. 查看所有的表

选择数据库后，才能查看表

```
show tables;
```

4. 删除表

删除表必须在数据库中进行删除

```
drop table [if exists] `表名`
```

5. 显示建表结构

```
desc `表名`;
describe `表名`;
```

6. 修改表

```
-- 修改表的名称
alter table `old_name` rename `new_name`;
-- 移动表 到指定的数据库
alter table `表名` rename to 数据库名.表名;
```

7. 修改字段

```
-- 增加一个新的字段
alter table `表名`  add `字段名` 数据类型 [属性];
-- 增加一个新的字段, 并放在首位
alter table `表名`  add `字段名` 数据类型 [属性] first;
-- 增加一个新的字段, 并放在某一个字段之后
alter table `表名`  add `字段名` 数据类型 [属性] after 指定字段;

-- 修改字段的属性
alter table `表名` modify `字段名` 数据类型 [属性];

-- 修改字段的名称
alter table `表名` change `原字段名` `新的字段名` 数据类型 [属性];
```

```
-- 修改字段的位置
alter table `表名` change `原字段名` `新的字段名` 数据类型 after `指定字段`;

-- 删除字段
alter table `表名` drop `字段名`;
```

8. 复制表

1. 先在创建一个表，并在表中插入一些数据

```
/* 创建 abc 表*/
create table abc(
    id int primary key auto_increment comment '主键',
    username char(32) not null comment '账户',
    password char(32) not null comment '密码'
);

/* 插入两条数据 */
insert into abc values(null, 'tom', md5(123456)), (null, 'bob',
md5(123456));
```

2. 复制表，并且复制数据
    - 执行:

      ```
      create table `复制表的名称` select * from `原表名`;
      ```

    - 特点:
        - 完整的复制一个表，既有原表的结构，又有原表的数据
        - 表内字段的属性会丢失，主键的自增等特性不复存在，新插入数据时会有问题
        - **最好不要使用这种方式复制**
3. 仅复制表结构, 不复制数据
    - 执行:

      ```
      create table `复制表的名称` like `原表名`;
      ```

    - 特点: 复制后的表结构与原表完全相同，字段的属性与原表也完全一致。但是里面没有数据，是一张空表
    - 如果需要，数据可以单独复制

      ```
      insert into `复制表的名称` select * from `原表名`;
      ```

七、CURD 语句的基本使用

对表中数据的操作一般分为四类, 常记做 "CURD":

- C: 创建（Create）
- U: 更新（Update）
- R: 读取（Retrieve）
- D: 删除（Delete）

1. INSERT 插入

完整的 insert 语句为:

```
INSERT INTO `表名` (`字段1`, `字段2`, ...) VALUES (`值1`, `值2`, ...);
```

其中的 `INTO` 在 MySQL 数据库中可以省略, 但在某些数据库中必须要有。

```
-- 一次插入一行
insert into `表名` set `字段`=值, `字段`=值;

-- 按照指定字段, 一次插入多行
insert into `表名` (字段1, 字段2 ...) values (值1, 值2, ...), (值1, 值2, ...);

-- 指定全部字段, 一次插入多行
insert into `表名` values (null, 值1, 值2, ...), (null, 值1, 值2, ...);
```

2. SELECT (查询)

```
-- 通过 * 获取全部字段的数据
select * from `表名`;

-- 获取指定字段的数据
select `字段1`, `字段2` from `表名`;
```

3. UPDATE (更新)

```
-- 修改全表数据
update `表名` set `字段1`=值, `字段2`=值;

-- 使用 where 修改满足条件的行
-- where 类似于 if 条件, 只执行返回结果为 True 的语句
update `表名` set `字段1`=值, `字段2`=值 where `字段`=值;
update `表名` set `字段1`=值, `字段2`=值 where `字段`=值 and `字段`=值;
```

4. DELETE (删除)

```
-- 删除表中的所有数据 (逐行删除)
delete from `表名`;
-- 清空全表 (一次性整表删除)
truncate `表名`

-- 使用 where 修改满足条件的行
delete from `表名` where `字段` = 值;
delete from `表名` where `字段` in (1, 2, 3, 4);
```

八、小总结

- 增
  - **数据库**:

```
create database `库名`;
```

- **表**:

```
create table `表名`;
```

- **字段**:

```
alter table `表名` add `字段名` 类型 [属性];
```

- **数据**:

```
insert into `表名`;
```

- 删
  - **数据库**:

```
drop database `库名`;
```

  - **表**:

```
drop table `表名`;
```

  - **字段**:

```
alter table `表名` drop `字段名`;
```

  - **数据**:

```
delete from `表名` where ...;
```

- 改
  - **数据库**:

```
alter database `库名` ...;
```

  - **表**:

```
alter table `表名` ...;
```

  - **字段**:

```
alter table `表名` modify | change ...;
```

  - **数据**:

```
update `表名` set ...;
```

- 查

- **数据库**:

```
show databases [like ...];
```

- **表**:

```
show tables [like ...];
```

- **字段**:

```
desc `表名`;
```

- **数据**:

```
select * from `表名` where ...;
```