数据库入门

一、数据库及其发展史

数据库是用来存储数据的, 数据是不是直接存储在数据库中?

不是的, 数据库中还有一个结构, 叫做表, 表中中存储的才是数据

我们要学习的: 数据库的连接, 数据库的创建, 表的创建, 数据的管理, 优化, 便捷操作, 关系的理解

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 层次模型

1. 优点:

   查询分类的效率比较高

2. 缺点:

   1. 没有导航结构, 导致分类困难
   2. 数据不完整

注意:数据的不完整, 如果不能准确的分辨两条数据有什么不同, 称之为失去了'数据的完整性'

3 网状模型

网状模型: 没有解决导航问题, 解决了数据完整性问题。

4 关系模型

现在的主流数据库都是关系模型的.

特点:

1. 每张表都是独立的, 没有导航结构
2. 表于表之间会建立公共字段, 也就将两张表之间建立了关系

注意: 公共的字段名可以不一样, 但是数据类型必须相同(数据类型相同的不一定是公共字段), 两个字段的含义必须也要一致.

关系型数据库, 解决了数据的完整性, 也解决导航问题, 但是带来的是低效率.

NOSQL(非关系型数据库): MongoDB, Redis

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。

三、Linux数据库的开启和连接

安装数据库

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

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`

连接数据库

各个 Linux 系统连接数据库都一样

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)

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'@'主机' IDENTIFIED BY "密码" WITH GRANT
OPTION;

-- 刷新使权限生效
flush privileges;
```

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

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 mysqld`

四、SQL语言概览

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

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。

关系型数据库

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

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. 字符编码不指定默认 utf8mb4
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 varchar(65535) comment '密码' not null
) engine=myisam charset=utf8mb4;
```

备注:

- 字符集如果不指定, 默认继承库的字符集.
- engine 默认innodb

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`;

-- 修改表的引擎
alter table `表名` engine = innodb|myisam;

-- 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'
   ) engine=myisam;

   /* 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
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相同，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;
```

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);
```