4~6天课程目录

第四天（mysql）

MySQL数据库服务器、数据库和表的关系

数据在数据库中的存储方式

> 数据库中的列也称之为字段或域

SQL语言

创建数据库

```
CREATE DATABASE [IF NOT EXISTS] db_name
[create_specification[,create_specification] ...]
create_specification: 7
[DEFAULT] CHARACTER SET charset_name | [DEFAULT] COLLATE
collation_name

练习：
创建一个名称为mydb1的数据库。
create database mydb1;
创建一个使用utf8字符集，并带校对规则的mydb3数据库。
create database mydb3 character set utf8 collate utf8_bin;
```

查看、删除数据库

```
显示数据库语句
SHOW DATABASES;
显示数据库创建语句
SHOW CREATE DATABASE db_name;
数据库删除语句
DROP DATABASE [IF EXISTS] db_name ;
```

修改数据库

```
ALTER DATABASE [IF NOT EXISTS] db_name
[alter_specification [, alter_specification] ...]
alter_specification:
[DEFAULT] CHARACTER SET charset_name | [DEFAULT]
COLLATE collation_name
```

选择数据库

```
use db_name;
查看当前使用的数据库
select database();
```

创建表(基本语句)

```
CREATE TABLE table_name
(
field1 datatype,
field2 datatype,
field3 datatype
)[character set 字符集][collate 校对规则]
```

MySQL常用数据类型

字符串型

> VARCHAR(20) 0~255
>
> CHAR(2) 0~255
>
> > varchar在存储数据的时候数据的长度不是固定的，可以在指定的范围内 存储任意长度的数据。varchar在读取数据的时候效率较低。 char在存储数据的时候数据的长度是固定的，尽管输入的数据没有达到 指定的长度也会占用数据库中的指定的长度。char读取数据的时候效率较高。

大数据类型

> BLOB
>
> TEXT

数值型

> TINYINT
>
> SMALLINT
>
> INT
>
> BIGINT
>
> FLOAT
>
> DOUBLE

逻辑型

> BIT

日期型

> DATE
>
> TIME
>
> DATETIME
>
> TIMESTAMP

定义单表字段的约束

定义主键约束

> primary key:不允许为空，不允许重复

> 删除主键：alter table tablename drop primary key
>
> 主键自动增长 ： auto_increment

定义唯一约束

> unique
>
> > name varchar(20) unique

定义非空约束

> not null
>
> > salary double not null

查看表信息

查看表结构

> desc tabName

查看当前所有表

> show tables

查看当前数据库表建表语句

> show create table tabName

修改表

使用 ALTER TABLE 语句追加, 修改, 或删除列的语法.

```
ALTER TABLE table_name ADD column datatype [DEFAULT expr]
[, column datatype]..;
ALTER TABLE table_name MODIFY column datatype [DEFAULT expr]
[, column datatype]...;
ALTER TABLE table_name DROP column;
```

修改表的名称:

> rename table 表名 to 新表名;

修改列的名称:

> ALTER TABLE table change old_column new_column typefiled;

修改表的字符集:

> alter table user character set utf8;

删除表

> drop table tabName;

第五天（mysql）

查询的顺序

> from--where--group by--having--select--order by

插入数据 create

```
INSERT INTO table_name [(column [, column...])] VALUES (value [,
value...]);

3.1.1
insert into table_name values(column_value,column_value...);
3.1.2
insert into table_name(id,name) values(null,'mxb');
```

插入中文出现乱码

> 乱码产生的原因是cmd窗口使用gbk字符集，数据库服务器使用utf-8字符集，两者不统一产生的。由于修改cmd窗口字符集十分繁琐，所以选择修改数据库的字符集。

> 1.在mysql客户端中输入 set names gbk;--临时修改，只在当前窗口内有效。 2.修改mysql根目录下的my.ini文件，将default-character-set=utf8改为default-character-set=gbk.重新启动mysql服务。--永久修改，修改过后，每一打开的客户端都是gbk字符集。

更新数据 update

> UPDATE语法可以用新值更新原有表行中的各列。

```
UPDATE tbl_name
SET col_name1=expr1 [, col_name2=expr2 ...]
[WHERE where_definition]

 练习：在上面创建的employee表中修改表中的纪录。
 将所有员工薪水修改为5000元。
 update employee set salary = 5000;
 将姓名为'zs'的员工薪水修改为3000元。
 update employee set salary = 3000 where name = 'caoyang';
 将姓名为'lisi'的员工薪水修改为4000元,job改为ccc。
update employee set salary = 4000,job='ccc' where name = 'lishuai';
将'wu'的薪水在原有基础上增加1000元。
update employee set salary = salary + 1000 where name = 'piaoqian';
```

删除数据 delete

> 使用 delete语句删除表中数据。

```
delete from tbl_name [WHERE where_definition]
```

> Delete语句不能删除某一列的值（可使用update）

```
update table_name set 字段名='';
```

> 使用delete语句仅删除记录，不删除表本身。如要删除表，使用drop table语句

```
drop table table_name;
```

查询数据 select

基本查询语句

> select * from table_name;

distinct 去重

> select distinct *|column_name from table_name;

在select语句中可使用表达式对查询的列进行运算

```
SELECT * |{column1 | expression, column2 | expression, ..}
FROM table;
```

在select语句中可使用as语句

> SELECT column as 别名 from 表名;

使用where子句，进行过滤查询

在where子句中经常使用的运算符

比较运算符

> `> < <= >= = <>`
>
> > 大于、小于、大于(小于)等于、不等于

between ...and...

> 显示在某一区间的值

in(set)

> 显示在in列表中的值，例：in("计算机系","英语系")

like ‘张pattern’

> 模糊查询%_

is null

> 判断是否为空 select * from user where id is null

ifnull(原值,替代值)

> 如果原值为null，则使用代替值 select ifnull(score,0) from exam;

逻辑运算符

and

> 多个条件同时成立

or

> 多个条件任一成立

not

> 不成立，例：where not(salary>100);

使用order by 子句排序查询结果。

> Asc 升序（默认）、Desc 降序 ORDER BY 子句应位于SELECT语句的结尾。

聚集函数(count,SUM,AVG,MAX/MIN)

count

count(列名)返回某一列，行的总数

> Select count(*)|count(列名) from tablenam [WHERE where_definition]

SUM

Sum函数返回满足where条件的行的和

> Select sum(列名){,sum(列名)...} from tablename[WHERE where_definition]

AVG

AVG函数返回满足where条件的一列的平均值

> Select avg(列名){,avg(列名)...} from tablename [WHERE where_definition]

MAX/MIN

> Select max(列名)from tablename [WHERE where_definition]

分组操作(group by)

```
SELECT column1, column2. column3..
FROM table
group by column having ...
```

第六天（JDBC，连接池）

详见第六天代码整理