SQL

数据库

创建数据库

1. 脚本创建

```
 #创建一个名称为testdb01的数据库
create database testdb01
on primary(
     -- 实体逻辑文件名称为testdb01
    name = 'testdb01',
     -- 创建在D:\testdb\testdb01.mdf 目录下
    filename = 'D:\testdb\testdb01.mdf',
    size = 10MB,
    maxsize = 100MB,
    filegrowth = 5MB
)
log on(
    name = 'testdb01_log',
    file name = 'D:\testdb\testdb01_log.ldf',
    size = 10MB,
    maxsize = 100MB,
    filegrowth = 5MB
)
```

2. GUI创建

- 数据库-> 右键->新建数据库

数据库修改

1. GUI修改

- 展开数据库->右键单击要修改的数据库->属性

2. 脚本修改

```
 #要修改的数据库为TESEDB01
ALTER DATABASE TESTDB01
 #TESTDB01名称修改为TESTDB02
MODIFY NAME = TESTDB02;

ALTER DATABASE TESTDB
 -- 修改文件
MODIFY FILE(
    NAME = 'TESTDT',
    SIZE = 20MB,
    MAXSIZE = 80MB,
    FILEGROWTH = 10MB
);
 -- 查看是否修改成功
```

```
EXEC SP_HELPDB TESTDB;

```

数据库删除

1. GUI删除

数据库->右键->任务->分离连接->删除

2. 脚本删除

```
 -- DROP 删除类型 名称1, 名称2;
DROP DATABASE Sales, NewSales;
```

如果当前数据库正在使用

```
use master
DROP DATABASE Sales
```

数据库的备份

1. 差异备份

只备份上一次数据与这一次备份的数据差集

2. 完整备份

右键数据库->任务->备份

数据库的还原

右键数据库->还原数据库->选择备份文件->还原

数据库的分离

1. 断开与当前数据库的连接
2. 右键要分离的数据库->任务->分离

数据库的附加

右键数据库->附加->选择要附加的.mdf文件

数据类型

1. 数字类型

数据类型	大小
bigint	8bit
int	4bit
smallint	2bit
tinyint	1bit
float	取决于数据类型

2.时间类型

数据类型	输出
time	时：分：秒.精确到小数点后7位
data	年-月-日
smalldatatime	年-月-日 时：分：秒
datatime	年-月-日 时：分：秒.小数点后三位
datatime2	年-月-日 时：分：秒.小数点后七位

3.字符串类型

一定要跟长度

类型	长度
char(n)	固定长度为n，n必须为1到8000
varchar(n)	可变长度，n用于定义字符串长度
nchar(n)	固定长度的Unicode字符，n用于定义长度，必须为1到4000之间的值
nvarchar(n)	可变长度的Unicode字符，n用于定义长度，可以为1到4000之间的值

表

表的相关数据

1. 字段：字段是一个事物（对象）的某一个静态特征（属性）
2. 记录：记录（对象）是字段的组合，表示一个具体的事物（对象）
3. 表：表是记录的组合，表示同一类事物（对象）的集合
4. 表、字段、记录的关系：字段是记录的属性，记录是事物本身，表是记录的集合
5. 列：列是字段的另一种称谓
6. 属性：字段的另一种称谓
7. 元组：记录的另一种称谓
8. 主键：（唯一识别码）唯一标识表的属性（防止表冗余）
9. 外键：主表的主关键字为从表的外键

10. 外键表：有外键的表称为外键表
11. 主键表：与外键表具有相同字段的表，外键表的属性来源于主键表
12. 约束：限制，保证数据合理

创建表

1. 图形界面创建

数据库->表->新建->表

2. 脚本创建

```
create table userinfo2(
    ID int primary key not null,
    name varchar(10) not null,
    age int null,
)
```

修改表结构

1. 更改字段类型长度

```
alter table 表
alter column 字段名 类型的长度 -- varchar(60)
```

2. 更改字段类型

```
alter table 表
alter column 字段名 更改后的类型
```

3. 添加是否为空约束

```
alter table 表
alter column 字段名 数字类型 not null
```

4. 设置主键

```
alter table 表
add constraint 主键名 primary key(字段名)
```

5. 更改字段名

`exec sp_rename '表名.字段名', '更改后的字段名','column'`

6. 添加字段名

```
alter table 表
add 字段名 字段类型 default null
```

7. 删除表

`drop table [表名1,表名2 ]`

8. 全部操作

```sql
-- 修改数据长度
alter table userinfo
alter column name varchar(100);

-- 修改字段名
exec sp_rename 'userinfo.userage', 'age','column'

-- 修改数据类型
alter table userinfo
alter column age float -- 默认为空

-- 修改限定
alter table userinfo
alter column age int not null;

-- 添加主键
alter table userinfo
add constraint KID primary key(ID);

-- 添加字段名
alter table userinfo
add sex varchar(10) not null;

-- 删除表
drop table userinfo2
```

主关键字（主键）

定义：是表中的一个或多个字段，它的值用于唯一地标识表中的某一条记录（使其不重复），***一个表只有一个主关键字（主键）***，主键可以由一个字段或多个字段组成 （单字段主键、多字段主键）

1. 添加主键

```sql
alter table 表名
add constraint 主键名 primary key(字段名)
```

2. 删除主键

```sql
alter table 表名
drop constraint 主键名
```

外关键字（外键）

1. 定义：外键表示两个关系之间的相关联系，以另一个关键的外键作主关键字的表被称为主表，具有此外键的表被称为主表，具有此外键的表被称为主表的从表
2. 目的：保存数据一致性，完整性，控制存储在外键表中的数据
3. 添加外键

```sql
alter table 从表名
add constraint 外键名 foreign key(字段名) reference 主表名(字段名)
```

4. 删除外键

```
alter table 从表名
drop constraint 外键名
```

建立关系（主键外键）

```
alter table sys_user_role
add constraint FK_user_role foreign key(RoleSysNo) referfences
sys_role(sysno)

```

新增表记录

1. 脚本

- 插入单行数据

```
insert into "表格名"("栏位1","栏位2",...)
values("值1","值2",...)
-- 一个括号对应一笔记录
-- 栏位和值一一对应
```

- 插入多行数据

```
insert into "表格名"("栏位1","栏位2",...)
values("值1","值2",...),("值1","值2"...),("值1","值2")
```

- 从其他表copy数据

```
insert into "表格1"("栏位1","栏位2",...)
select "栏位3","栏位4",...
-- 栏位同样一一对应
from "表格2"
```

2. 代码

```
-- 星号代表所有字段
select * from [dbo].[userinfo]

insert into userinfo (userid, username, email)
values ('zhang','张三','zhang@qq.com')
select * from [dbo].[userinfo]


insert into userinfo(userid, username, email)
values('zhang01','张三','zhang@qq.com'),('zhang02','张三','zhang@qq.com'),
('zhang03','张三','zhang@qq.com')
```

查询表记录

1. 脚本

- 查询

```
select 字段名
from 表
```

- 给表取别名然后查询

```
select 别名.字段名... from 表名 as 别名
```

- 去重查询(不影响实际数据)

```
select distinct 字段名
form 表
```

- 查询前n行

```
select top 行数 字段名
from 表
```

- 选择某个值的字段

`where 字段 = 值`

2. GUI

右键表->编辑前1000行

3. 代码

```
select *
from userinfo

select userid, username
from userinfo

select distinct *
from userinfo

select top 3 *
from userinfo
```

修改表记录

1. 脚本

```
update 表名
set 字段 = 值
```

2. GUI

右键表->编辑前200行->编辑

删除数据表记录

1. 脚本

```
delete from 表名
where 字段 = 值
```

2. GUI

右键表->编辑->右键->删除

条件限制（where）

1. 精确限制条件

`where 字段 = 值`

2. 模糊限制条件

```
-- 以值结尾的属性
where 字段 like'%值'

-- 以值起始的属性
where 字段 like'值%'

-- 带有值的属性
where 字段 like'%值%'
```

BETWEEN语法

用于限制条件表达式

```
-- 查询[起始,结束]的值
where 字段名 bewtween 起始值 and 结束值

-- 查询![起始，结束]的值
where 字段名 not between 起始值 and 结束值
```

子查询（IN表达式）

用于限制条件表达式

```
-- 查询字段为值1，值2，值3的数据
where 字段名 in(值1,值2,值3,...)

-- 查询字段不为值1，值2，值3的数据
where 字段名 not in(值1,值2,值3,...)
```

```
-- 通过子查询来限制主查询
select * from 表1
where 字段名 in (select 字段名 from 表2)
```

子查询（EXISTS）

exists 返回的是一个bool类型

```
select a.StudentNo, a.StudentName, a.Age, a.Sex from students as a
-- 返回false不显示 返回true显示
where exists(select * from student_lesson b where a.StudentNo = b.StudentNo)
```

返回结果排序

```
[查询结果]
order by 字段1 方式1,字段2 方式2,...
--方式：asc升序，desc降序
```

关联查询

1. 交叉关联

返回两个表中联结字段相等的行

```
select * from 表1 as a
inner join 表2 as b
on a.字段 = b.字段
-- 同时返回表1 表2满足关联关系的数据值
```

2. 左关联

返回包括左表中所有的记录和右表中联结字段相等的记录

```
select * 表1 as a -- 左表
left join 表2 as b -- 右表
on a.字段 = b.字段
```

3. 右关联

返回包括右表中所有记录和左表中国联结字段相等的记录

```
select * from 表 as a -- 左表
right join 表2 as b -- 右表
on a.字段 = b.字段
```

4. 多表关联

```
select a.字段1, a.字段2,c.字段3 from 表1 as a
inner join 表2 as b
on a.字段1 = b.字段1
inner join 表3 as c
on b.字段2 = c.字段2
```

关联关系也可以在 inner join 行处用 and 附加条件

函数

聚合函数

返回的是一个列

给列取别名：`列 as 列名`

1. AVG()

返回组中各值的平均值，忽略null值。计算字段类型必须用为数字

```
select avg(字段)
from 表名
```

2. SUM()

返回表达式中所有值的和。其中忽略null值。同样只能用于数字

```
select sum(字段)
from 表
```

3. MIN()

返回表达式的最小值，忽略null值

```
select min(字段)
from 表
```

4. MAX()

返回表达式的最大值，忽略null值

```
select max(字段)
from 表
```

5. COUNT()

返回组中的项数，忽略null值

```
select count(字段)
from 表()
```

6. COUNT_BIG()

与count()用法一样，返回值的范围比count()大

7. LEN()

返回指定字符串表达式的字符数，不包含尾随空格

```
select len(字段)
from 表
```

8. DATALENGTH()

返回字节数

```
select datalength(字段)
from 表
```

字段合并

```
select 字段1 + 字段2 as 列名 from 表
-- 相当于字符串拼接
```

随机数产生

1. RAND()

随机产生一个小数

`select rand()`

2. FOOR()

向下取整

3. CEILING()

向上取整

4. 代码

```
select floor(rand() * N)
select ceiling(rand() * N)
```

当前时间获取

1. GETDATE()

返回当前数据库系统时间值，返回类型为datetime

`select getdate()`

2. GETUTCDATE()

返回国际标准时间，返回类型为datetime

`select getutcdate()`

3. CONVENT()

把日期转换为新数据类型，用不同格式显示日期/时间数据

不同的格式对应不同的格式ID

`select convent (目标数据类型(长度),日期,格式ID)`

时间计算

相差/增加部分 为 second, minute , hour , day, month, year

1. DATEDIFF()

返回两个日期之间的差

`select datediff(相差部分, 开始时间, 结束时间)`

2. DATEADD()

在日期中添加或减去指定的时间间隔

`select DATEADD(增加的部分, 增加的数值（可正可负）, 时间)`

获取日期的部分值

3. DATEPART()

datepart()函数用于返回日期/时间的单独部分，类型为int

`select datepart(返回值（yy或mm或dd）, 时间)`

4. DATENAME()

datename()返回日期/时间单独部分，类型为varchar()

`select datename(返回值 （yy或mm或dd）,时间)`

5. DAY(), MONTH(), YEAR()

返回天数，月份，年份

字符串的匹配查找

可支持条件限制

1. CHARINDEX()

返回字符或者字符串在另一个字符串的起始位置，若没查到则返回0

`select charindex(字符串1, 字符串2, 起始位置（非必要参数）)`

起始位置是指被查找字符串的起始位置（字符串2）

2. PARTINDEX()

返回字符或者字符串在另一个字符串或者表达式的起始位置，模糊查询

`select partindex(%字符串1%, 字符串2, 起始位置（非必要参数）)`

```
-- 查询字符串1是否为字符串2的子串
select partindex(%字符串1%, 字符串2)
-- 查询字符串2是否以字符串1开头
select partindex(字符串1%, 字符串2)
-- 查询字符串2是否以字符串1结尾
select partindex(字符串1%, 字符串2)
```

修改字符串

1. STUFF()

stuff()用于删除指定长度的字符，并可以在指定的起点处插入另一组字符，返回类型为字符串

`select stuff(列名, 开始位置, 操作长度, 替代字符串)`

```
-- 从开始位置，对操作长度个字符进行操作

-- 删除efg
select stuff('abcdefg', 5,3,'')

-- 替换ef
select stuff('abcdefg', 5,2,'aaa')
```

2. SUBSTRING()

从开始位置截取指定长度的字符串并返回

`select substring(字符串, 开始位置, 截取长度)`

3. LEFT(), RIGHT()

分别从左边、右边返回指定长度个数的字符，指定长度不能为负数

`select left(字符串, 长度)`

`select right(字符串, 长度)`

4. LTRIM(), RTTIM()

分别删除起始、尾随空格

`select ltrim(字符串)`

`select rttim(字符串)`

```
-- 删除起始、尾随空格
select *, rtrim(ltrim(StudentName)) as NewStudentName from Students
```

5. UPPER(), LOWER()

把字符串分别转换为大写、小写

`select upper(字符串)`

`select lower(字符串)`

6. REPLACE()

用一个字符串替换出现的***所有***指定字符串值

`select replace(被替换串, 被替换值, 替换值)`

7. REPLICATE(), SPACE()

replicate()：返回以指定次数重复的字符串

`select replicate(字符串, 指定次数)`

space()：返回指定个数的空格

`select space(指定个数)`

8. REVERSE()

反转字符串

`select reverse(字符串)`

9. CAST()

用于将某种数据类型的表达式显式转换为另一种数据类型

`cast(字符串 as 转换类型)`

10. CASE()

把满足条件的表达式转换为对应的结果

1. 简单case函数

```
case 字段
    when 值1 then 操作1
    when 值2 then 操作2
    when 值3 then 操作3
    else 操作4
end

```

1. 搜索函数

```
case
    when 字段 = 值1 then 操作1
    when 字段 = 值2 then 操作2
    when 字段 = 值3 then 操作3
    else 操作4
end
```

应用：对数据进行分类

```sql
select *,
case
    when score > 90 then '优'
    when score > 80 and socre < 90 then '良'
    when score > 70 and score < 80 then '中'
    else '差'
end as grade
from score
```

搜索函数中也可以嵌套in表达式

`when 字段 in(值1, 值2, 值3,..) then 操作`