数据库

数据库的基本概念及操作

概念

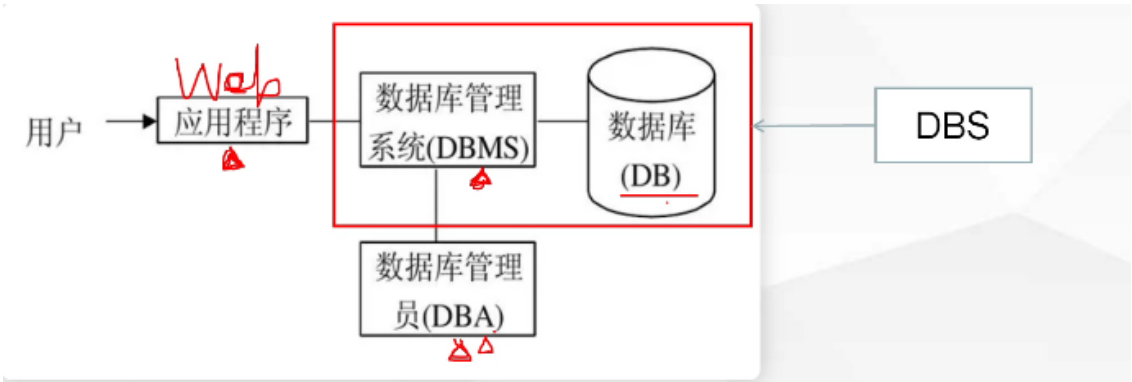
- 数据库的定义

  数据库(Database,简称DB)是**长期储存**在计算机内、**有组织**的、**可共享**的**大量**数据的集合。

- 数据库的基本特征

  1. 数据按一定的数据模型组织、描述和储存
  2. 可为各种用户共享
  3. 冗余度较小
  4. 数据独立性较高
  5. 易扩展

- 数据(Data):的定义
  - 定义：描述事物的符号记录，是数据库中存储的基本对象
  - 种类：文字、图形、图象、声音。
- 数据的记录：计算机中表示和存储数据的一种格式或一种方法
  - 例子：(李明，男，1998，江苏，计算机系，2017)
  - 语义：学生姓名、性别、出生年月、籍贯、所在系别、入学时间
- 数据库管理系统
  - 什么是DBMS(数据库管理系统-数据库的核心)
    - 位于用户与操作系统之间的一层数据管理软件。
    - 是系统软件，是一个大型复杂的软件系统
- DBMS的用途
  - 科学地组织和存储数据、高效地获取和维护数据
- 数据库系统(Database System,简称DBS)
  - 定义：是指在计算机系统中引入数据库后的系统构成。常常把**数据库系统**简称为数据库。
  - 构成：由数据库、数据库管理系统（及其开发工具）、应用程序、
    数据库管理员（和用户)构成。
  - 注：DBS(数据库系统)=DB(数据库)+DBMS(数据库管理系统)
- 数据库系统(DBS)数据库(DB)、数据库管理系统(DBMS)的关系



- 数据库在计算机系统中的位置

- 数据库系统发展
  - 人工管理阶段(40年代中-50年代中)
  - 文件系统阶段(50年代末-60年代中)
    - 数据共享性差，冗余度大
    - 数据独立性差
  - 数据库系统阶段(60年代末-现在)
    - 数据共享性高，冗余度低且容易扩充
    - 数据独立性高

SQLServer2012简介

- 版本
  - √企业版(Enterprise)
  - √商业智能(Business intelligence)
  - √标准版(Standard)
  - √Web版
- SQLServer2012数据库相关概念
  - 数据库文件和日志
    1. 主数据文件：包含数据库的启动信息，用户数据和对象存储在此文件中，扩展名为.mdf
    2. 次要数据文件：可选，扩展名为.ndf
    3. 事务日志文件：保存用于恢复数据库的日志信息，扩展名.ldf

       提示：虽然SQL Server2012不强制这3种类型文件必须使用带mdf、ndf和ldf扩展名，但使用它们指出文件类型是个良好的文件命名习惯。
       注：以上几种文件都放在文件组中，主文件和日志文件必须有，其中主文件只能有一个，其他类型文件可以有多个

数据库操作

- 创建数据库 语法

```
CREATE DATABASE 数据库名;
on primary  -- 主数据文件  只能有一个
(
name = mydb,    -- 名字 -- 自定义最少有名字
filename ='D:\data\mydb.mdf',  -- 位置 文件名
size=5,  -- 初始大小
maxsize= 20,  -- 最大大小
filegrowth =10% -- 增长方式 百分比 绝对值
```

```
),
( -- 次数据文件
name = mydb_sec,   -- 名字
filename ='D:\data\mydb_sec.ndf',  -- 位置文件名
size=5,  -- 初始大小
maxsize= 20,  -- 最大大小
filegrowth =10% -- -- 增长方式 百分比 绝对值
),
log on    -- 日志文件 最少一个
(
name = mydb_log, -- 名字
filename ='D:\data\mydb_log.ldf', -- 位置
size =2,  -- 大小
maxsize=10, -- 最大大小
filegrowth =1 -- 每次增长多少
)
```

- 例子

```
create database MyDb
on primary (
size=20,
name = MyDb,
maxsize=unlimited, -- unlimited 最大尺寸
filegrowth=10%,
filename='D:\EXE\BC\sql server\sql server
sl\MSSQL11.MSSQLSERVER\MSSQL\DATA\MyDb.mdf'
),
(
name=MyDb_sec,
size=20,
maxsize=200, -- unlimited 最大尺寸
filegrowth=10%,
filename='D:\EXE\BC\sql server\sql server
sl\MSSQL11.MSSQLSERVER\MSSQL\DATA\MyDb_sec.ndf'
)
log on
(
name=MyDb_log,
size=20,
maxsize=200, -- unlimited 最大尺寸
filegrowth=10%,
filename='D:\EXE\BC\sql server\sql server
sl\MSSQL11.MSSQLSERVER\MSSQL\DATA\MyDb_log.ldf'
)
```

- 修改数据库
- 修改数据库名称 语法

```
alter database 数据库名 modify name=数据库新名字
```

- 例子

```
alter database MyDb modify name=mydd
```

- 增加数据文件 语法

```
alter database 数据库名  add file(
name = mydb,    -- 名字 -- 自定义最少有名字
filename ='D:\data\mydb.mdf',  -- 位置 文件名
size=5,  -- 初始大小
maxsize= 20,  -- 最大大小
filegrowth =10% -- 增长方式 百分比 绝对值
)
```

- 例子

```
alter database mydb add file(add log file)( -- 次数据日志
name = mydb,    -- 名字 -- 自定义最少有名字
filename ='D:\data\\',  -- 位置 文件名
size=5,  -- 初始大小
maxsize= 20,  -- 最大大小
filegrowth =10% -- 增长方式 百分比 绝对值
)
```

- 修改文件（数据或日志）语法

```
alter database 数据库名 modify file(
name = mydb,    -- 名字 -- 自定义最少有名字
filename ='D:\data\mydb.mdf',  -- 位置 文件名
size=5,  -- 初始大小
maxsize= 20,  -- 最大大小
filegrowth =10% -- 增长方式 百分比 绝对值
)
```

- 例子

```
alter database mydb modify file(
name=mydb_data,
size=5MB
)
```

- 删除(移除)文件(数据或日志) 语法

```
alter database 数据库名 remove file 数据文件逻辑名
```

- 例子

```
alter database mydb remove file mydb_data
```

- 删除数据库 语法

```
drop database 数据库名
```

- 例子

```
drop database mydb
```

管理维护数据库

分离附加数据库

- 分离数据库 语法 （系统数据库master、empdb、moel不可分离）

```
execute sp_detach_db 数据库名
```

例子

```
execute sp_detach_db mydb
```

- 附加数据库 语法

```
execute sp_attach_db 数据库名 '文件位置(需要加文件名)'
```

例子

```
execute sp_attach_db mydb ,'D:\EXE\BC\sql server\sql server
sl\MSSQL11.MSSQLSERVER\MSSQL\DATA\MyDb_log.bak'
```

- 注：通过分离和附加数据库可以实现SQL Server数据库**文件存储位置的改变**（移植数据库）
- 备份 语法

```
backup database 数据库名 to disk = '文件位置(需要加文件名)'
```

例子

```
backup database mydb to disk = 'D:\EXE\BC\sql server\sql server
sl\MSSQL11.MSSQLSERVER\MSSQL\DATA\MyDb_log.bak'
```

- 还原 语法

```
restore  database 数据库名 from disk = '文件位置(需要加文件名)'
```

例子

```
restore  database 数据库名 from disk = 'D:\EXE\BC\sql server\sql server
sl\MSSQL11.MSSQLSERVER\MSSQL\DATA\MyDb_log.bak'
```

数据表操作

- 表和表结构：每个数据库包含了若干个表。表是SQL Server中最主要的数据库对象，它是用来**存储数据**的一种**逻辑结构。表由行和列组成**，因此也称为**二维表。**表是在日常工作和生活中经常使用的一种**表示数据及其关系**的形式
- 下面简单介绍与表有关的几个概念：
  1. 表结构。组成表的各列的名称及数据类型，统称为表结构。
  2. 记录。每个表包含了若干行数据，它们是表的"值"，表中的一行称为一个记录。

3. 字段。表中的一列称为字段。例如，表3.1中表结构为（学号，姓名，性别，出生时间，专业，总学分，备注），包含7个字段，由5个记录组成。
4. 空值。空值(NULL)通常表示未知、不可用或将在以后添加的数据。若一个列允许为空值，则向表中输入记录值时可不为该列给出具体值；而一个列若不允许为空值，则在输入时必须给出具体值。
5. 关键字。若表中记录的某一字段或字段组合能唯一标识记录，则称该字段或字段组合为候选关键字(Candidate key)。若一个表有多个候选关键字，则选定其中一个为主关键字（Primary key),也称为主键。

- 创建表 语法

```
create table 表名(
字段1 字段1类型 列级完整性约束条件,
字段2 字段2类型 列级完整性约束条件,
字段3 字段3类型 列级完整性约束条件,
...
字段n 字段n类型 列级完整性约束条件
)
```

- 注：列级完整性约束条件如下：

1. PRIMARY KEY:指定该字段为主键（不为空且唯一）
2. NULL/NOT NULL:指定的字段允许/不允许为空，如果没有约束条件，则默认为NULL(列级约束)
3. UNIQUE:指定字段取值唯一，即每条记录的指定字段值不能重复 (允许列中有一个空值)
4. DEFAULT<默认值>：指定设置字段的默认值。（列级约束）
5. CHECK<条件表达式>：对输入值检验，拒绝接受不满足条件的值。只看非空约束 （不是null的）
6. identity(1,1)自增长（第一个值,步长）不能直接添加数据自动增长

   如果想强制执行 set identity_insert 表名 on
7. foreign key 外键 注意字段类型必须一样

   例子

   ```
   foreign key reference 表名(字段)
   ```
8. 第二条和第四条是列级约束其他的是列级约束和表级约束都可以　　列级约束是 只能针对当列约束

   表级约束可以多列

- 例子

```
create table student(
 sno char(8) not null primary key,
 sanme char(10) not null,
 grender char(2) null default '男' check(grender='女' or grender='男'),
 sbirth date,
 email char(30) unique, -- 唯一约束
 major char(20),
 chedit int check(chedit>=0 and chedit<120),
 remark varchar(500)
)
```

- 注意：一个主键可以由多个字段构成

```
create table student(
 sno char(8) ,
 sanme char(10),
 grender char(2) null default '男' check(grender='女' or grender='男'),
 sbirth date,
 email char(30) unique, -- 唯一约束
 major char(20),
 chedit int check(chedit>=0 and chedit<120),
 remark varchar(500),
  primary key(sno,sanme)  --联合主键
)
```

- 修改数据表 语法

```
-- 增加属性
alter table 表名 add 新字段名称 数据类型 列级完整性约束条件
-- 修改属性的数据类型
alter table 表名 alter column 字段名称 新数据类型
-- 添加约束
alter table 表名 alter constraint 约束名 约束条件 (字段名称)
-- 删除约束
alter table 表名 drop constraint 约束名
-- 删除属性
alter table 表名 drop column 字段名称
```

- 例子

```
-- 增加属性
alter table student add sql char(30) null
-- 修改属性的数据类型
alter table student alter column sql char(20)
-- 添加约束
alter table student add constraint uq_stu_sq unique (sql)
-- 删除约束
alter table student drop constraint uq_stu_sq
-- 删除属性
alter table student drop column sql
```

- 删除表 语法

```
drop table 表名
```

- 例子

```
drop table student
```

关系的完整性

- 关系的完整性（最大限度地保证数据的**正确性**）
- 关系模型的完整性规则是对关系的某种约束条件。
- 关系模型中允许定义3类完整性约束：

1. √实体完整性

   实体完整性规则若属性A是基本关系R的主属性，则属性A不能取空值。
   例如：学生关系“学生学号，姓名，性别，专业号，年龄”中，“学号”
   为主码，则“学号”不能取空值。

2. √参照完整性

   学生、课程、学生与课程之间的多对多联系选修可以用如下3个关系表示。
   √学生（**学号**，姓名，性别，专业号，年龄）
   √课程（**课程号**，课程名，学分）
   √选修（**学号**，**课程号**，成绩）

3. √用户自定义的完整性

   用户自定义的完整性就是针对某一具体关系数据库的约束条件，它反映某一具体应用所涉及的数据必须满足语义要求。例如某个属性必须取唯一值、属性值之间应满足一定的函数关系、某属性的取值范围在0~100之间等。
   √例如，性别只能取“男”或“女”；学生的成绩必须在0~100之间。

操作数据表中的数据

插入

1. 插入单条记录
   - 语句格式

     ```
     insert into 表名 [列名清单] values 常量清单
     ```

     如果省略<**列名清单**>，则按<常量清单>顺序为**每个属性**列赋值，即每个属性列上都应该有值
   - 例子

     ```
     insert into student values ('20200101','张三','男','2001-06-09','666@QQ.COM','计算机系',100,'备注1')
     ```

   - 注：
     1. 表中不允许为空值的项必须输入
     2. 省略列名清单，则常量清单应于表中的列名顺序一致
2. 插入多条记录
   - 语句格式

     ```
     insert into 表明 [列名清单] values (常量清单1),(常量清单2),...
     ```

   - 例子

     ```
     insert into student values ('20200102','张三','男','2001-06-09','999@QQ.COM','计算机系',100,'备注1'),('20200103','萧炎','男','2001-06-09','66666@QQ.COM','计算机系',100,'备注1'),('20200104','刘6','男','2001-06-09','666999@QQ.COM','计算机系',100,'备注1'),
     ('20200105','张齐','男','2001-06-09','666999666@QQ.COM','计算机系',100,'备注1'),('20200106','钱多多','男','2001-06-09','999666999666@QQ.COM','计算机系',100,'备注1')
     ```

   - 小结

1. INSERT语句中的INTO可以省略
2. 如果某些属性列在表名后的列名表没有出现，则新记录在这些列上取空值
3. 如果没有指明任何列名，则新插入的记录必须在每个属性列上均有值
4. 字符型或日期型数据必须使用''（单引号）将其括起来
5. 常量的顺序必须和指定的列名顺序保持一致

修改

- 语句格式

```
update 表名 set 列名1=表达式1 [,列名2=表达式2] [where 条件表达式]
```

说明：

1. 如果不指定条件，则会修改表中所有记录
2. 如果要修改多列，则在SET语句后用”，”分割各修改子句
3.

- 例

```
update student set sanme ='刘德华' where sanme='张齐'
update student set chedit=60 where chedit<60
```

删除

- 语句格式

```
delete from 表名  [where 条件表达式]
```

说明：
当无WHERE（条件表达式>时将删除<表名>中所有记录，但是，该表结构还在，只是变为了空表

- 例

```
delete from student where sanme='刘德华'
```

单表查询

- 语句格式

```
SELECT all[DISTINCT][TOP N[PERCENT]],字段1,字段2,字段3..[AS 别名] FROM 表名;
```

- 说明

1. ALL:表示输出所有记录，包括重复记录。默认值为ALL。
2. DISTINCT:表示在查询结果中去掉重复值。
3. TOP N:返回查询结果集中的前W行。

   加[PERCENT]返回查询结果集中的前N%行。W的取值范围是0~100。

- 例

```
select * from student
select distinct *  from student
select top 6 *  from student
-- top % 计算结果向上取整数
select top 80 percent  * from student
select sno as '学号'  from student
-- 计算年龄现在减出生日期
select sanme ,year(getdate())-year(sbirth) from student
-- 分数提20%
select sno,chedit*1.2 AS 成绩 from student
```

- 单表有条件查询
- 语句格式

```
SELECT all[DISTINCT][TOP N[PERCENT]],字段1,字段2,字段3..[AS 别名] FROM 表名
where 条件;
```

- WHERE条件中的运算符

查询条件	运算符
比较运算符	=, <, >, <=, >=, !=
逻辑运算符	AND,OR,NOT
范围运算符	BETWEEN AND,NOT BETWEEN AND
列表运算符	IN,NOT IN
字符匹配符	LIKE,NOT LIKE
空值	IS NULL,IS NOT NULL

1. WHERE子句中可以使用逻辑运算符AND、OR和NOT,这3个逻辑运算符可以混合使用。
2. 在WHERE子句中使用BETWEEN关键字查找在某一范围内的数据，也可以使用NOT BETWEEN关键字查找不在某一范围内的数据。
3. 在WHERE子句中使用字符匹配符LIKE或NOT LIKE可以把表达式与字符串进行比较，从而实现对字符串的模糊查询。

   通配符% 表示0或者多个字符

   通配符_ 表示任意一个字符
4. 在WHERE子句中，如果需要确定表达式的取值是否属于某一列表值之一时，就可以使用关键字IN或NOT IN来限定查询条件。
5. 当数据表中的值为NULL时，可以使用IS NULL关键字的WHERE子句进行查询，反之要查询数据表的值不为NULL时，可以使用IS NOT NULL关键字。**注意无法用等于号（=）判断空**

- 例子

```
select * from student WHERE grender='男'
select * from student WHERE chedit>60
select * from student WHERE major='计算机系'
-- 查询计算机系女生的信息。
select * from student where major='计算机系' and grender='女'
```

```
-- 查询成绩在90分以上或不及格的学生学号和课号信息。
select * from student where chedit>90 or chedit<60
-- 查询非计算机
select * from student where not major='计算机系'
-- 查询成绩在60~70分之间含(60,70)的学生学号及成绩。
select sno,chedit from student where chedit between 60 and 70
-- 查询所有姓张的学生的个人信息。
select * from student where sanme like '张%'
-- 查询所有名字第2个字带三的学生的个人信息。
select * from student where sanme LIKE '_三%'
-- 查询所有姓张姓唐的学生的个人信息。
select * from student where sanme like'[张唐]%'
-- 查询软件和计算机的学生
select * from student where major in('软件','计算机系')
--查询缺少成绩的学生
select * from student where chedit is  null
-- 不为空
select * from student where chedit is not null
```

聚合函数

- SQL Server的聚集函数是综合信息的统计函数，也称为聚合函数或集函数，
  1. 包括计数、求最大值、求最小值、求平均值和求和等。
  2. 聚集函数可作为列标识符出现在SELECT子句的目标列或HAVING子句的条件中。
  3. 在SQL查询语句中，如果有GROUP BY子句，则语句中的函数为分组统计函数；否则，语句中的函数为全部结果集的统计函数。SQL提供的聚集
函数	说明
COUNT(*)	统计行的个数
COUNT(<列名>)	统计一列中值的个数
MAX(<列名>)	计算一列中值的最大值
MIN(<列名>)	计算一列中的最小值
SUM(<列名>)	计算一列中值的总和
AVG(<列名>)	计算一列中值的平均值

  注意：**聚集函数忽略空值** 所以用非空项查询， **除了count其他都不能*号**
- 例子

```
-- 查询学生总数。
select count(*) from student
--查询选修了课程的学生人数。
select count(distinct sno) from sc
--计算男学生平均成绩。
select avg(chedit) from student where grender='男'
--查询学生最高分和最低分。
select max(chedit),min(chedit) from student
```

-

数据库高级应用