



MySQL数据库基础语法

第二章

课程内容

- 1、MySQL的两大存储引擎
- 2、数据的完整性概述
- 3、四大SQL语句
- 4、库表操作---约束、数据类型
- 5、插入数据的SQL语句
- 6、修改，删除数据的SQL语句
- 7、查询数据的SQL基本语法
-

课程目标

- 了解MySQL两种存储引擎；
- 熟悉MySQL中的数据类型；
- 熟练掌握基本的SQL语句，如建表建库，对表中的数据进行增删改查；

表 - dbo.Students 表 - dbo.Course 摘要

▼ X

列名	数据类型	允许空
SName	char(10)	<input type="checkbox"/>
SCode	int	<input type="checkbox"/>
SAddress	nvarchar(50)	<input checked="" type="checkbox"/>
SGrade	float	<input checked="" type="checkbox"/>
SEmail	nvarchar(50)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

学员信息表

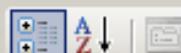
列名	数据类型	允许空
StudentID	int	<input type="checkbox"/>
CourseID	int	<input type="checkbox"/>
Score	smallint	<input checked="" type="checkbox"/>
ScoreID	int	<input type="checkbox"/>
		<input type="checkbox"/>

学员成绩表

列名	数据类型	允许空
CourseID	int	<input type="checkbox"/>
CourseName	nvarchar(50)	<input type="checkbox"/>

课程表

列属性



(常规)

(名称)	SName
长度	10
默认值或绑定	
数据类型	char
允许空	否

1.1 MyISAM和InnoDB存储引擎

MySQL提供了插件式（Pluggable）的存储引擎，存储引擎是基于表的，同一个数据库，不同的表，存储引擎可以不同。甚至同一个数据库表，在不同的场合可以应用不同的存储引擎。

使用MySQL命令即可查看MySQL服务实例支持的存储引擎。

```
show engines;
```

1.2 MyISAM和InnoDB存储引擎

1. InnoDB存储引擎的特点

- 支持外键 (Foreign Key)
- 支持事务 (Transaction)：如果某张表主要提供OLTP支持，需要执行大量的增、删、改操作 (insert、delete、update语句)，出于事务安全方面的考虑，InnoDB存储引擎是更好的选择。
- 最新版本的MySQL已经开始支持全文检索。

2. MyISAM存储引擎的特点

- MyISAM具有检查和修复表的大多数工具。
- MyISAM表可以被压缩
- MyISAM表最早支持全文索引
- 但MyISAM表不支持事务
- 但MyISAM表不支持外键 (Foreign Key)。

如果需要执行大量的select语句，出于性能方面的考虑，MyISAM存储引擎是更好的选择

1. 3 设置默认的存储引擎

MySQL5.6默认的默认的存储引擎是InnoDB。

使用MySQL命令

```
set default_storage_engine=MyISAM;
```

可以“临时地”将MySQL“当前会话的”存储引擎设置为MyISAM， 使用MySQL命令“show engines;”可以查看当前MySQL服务实例默认的存储引擎。



新梦想
NEW DREAM

2.1 再论数据完整性

可靠性



准确性



数据完整性

2.2 再论数据完整性

- 数据存放在表中
- “数据完整性的问题大多是由于设计引起的”
- 创建表的时候，就应当保证以后数据输入是正确的
——错误的数据、不符合要求的数据不允许输入

创建表：保证数据的完整性 = 实施完整性约束

2.2完整性包括...

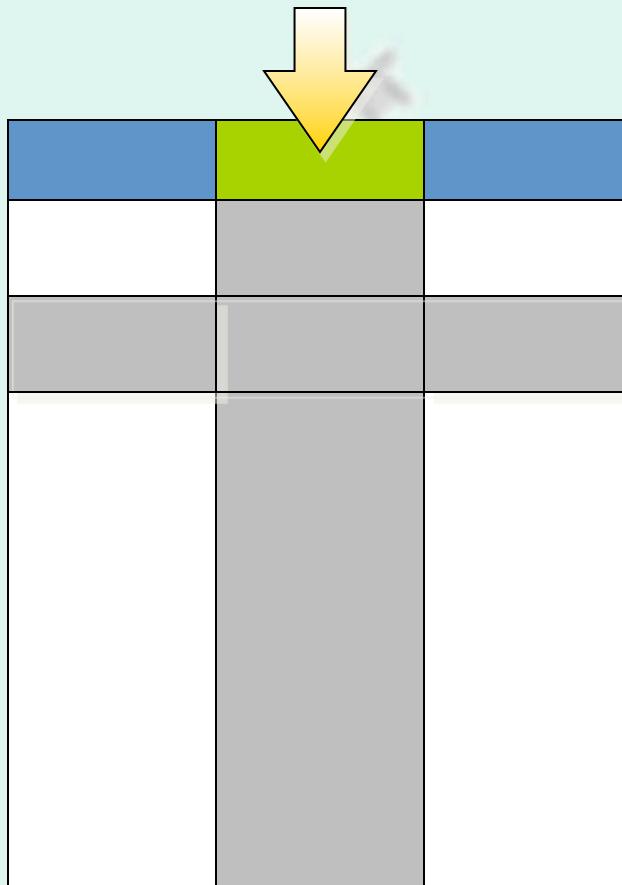
- 输入的类型是否正确?
 - 年龄必须是数字
- 输入的格式是否正确?
 - 身份证号码必须是18位
- 是否在允许的范围内?
 - 性别只能是”男”或者”女”
- 是否存在重复输入?
 - 学员信息输入了两次
- 是否符合其他特定要求?
 - 信誉值大于5的用户才能够加入会员列表
-

列值要求（约束）

整行要求（约束）

2.2完整性包括...

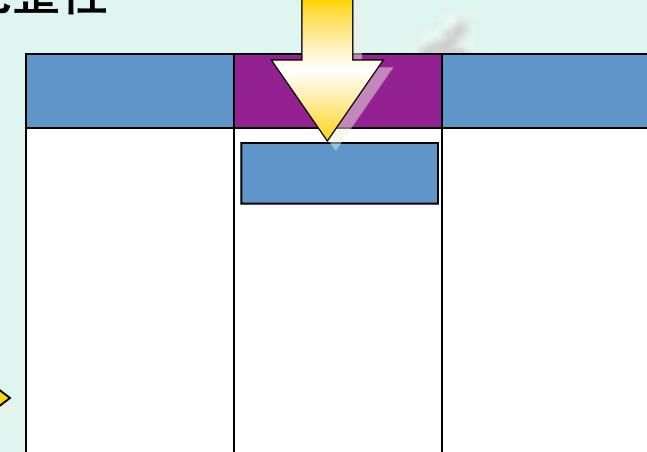
域完整性



实体完整性

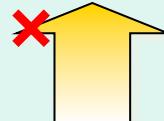
自定义完整性

引用完整性



2.3 实体完整性

学号	姓名	地址
0010012	李山	山东定陶	
0010013	吴兰	湖南新田	
0010014	雷铜	江西南昌	
0010015	张丽鹃	河南新乡	
0010016	赵可以	河南新乡	

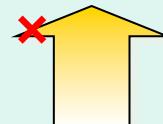


0010014 雷铜 江西南昌

约束方法：唯一约束、主键约束、标识列

域完整性

学号	姓名	地址
0010012	李山	山东定陶	
0010013	吴兰	湖南新田	
0010014	雷铜	江西南昌	
0010015	张丽鹃	河南新乡	
0010016	赵可以	河南新乡	



8700000000 李亮 湖北江门

约束方法：限制数据类型、检查约束、外键约束、默认值、非空约束

引用完整性

↓

学号	姓名	地址	...
0010012	李山	山东定陶	
0010013	吴兰	湖南新田	
0010014	雷铜	江西南昌	
0010015	张丽鹃	河南新乡	
0010016	赵可以	河南新乡	

科目	学号	分数	...
数学	0010012	88	
数学	0010013	74	
语文	0010012	67	
语文	0010013	81	
数学	0010016	98	

约束方法：外键约束

×

数学	0000012	98	
----	---------	----	--

使用图形界面完成建库的操作

1. 新建学生库
2. 新建3个表,改表名
3. 新增字段: 字段名称
4. 设置字段类型

注意表必须有一个字段才能保存

使用图形界面完成建库的操作

5. 设置是否为空

6. 设置主键

主键约束：不能为空，不能重复

唯一约束：不能重复，允许一条记录为空

一个表最多一个主键或者没有，但可以多个字段一起建主键

7. 将学生信息表中学号设置为自己增加

种子 增量 作用：自动取值，流水号

8. 设置默认值 地址设置为默认值

9. 表关系 在从表上建关系

主表----从表

students.scode-----score.studentid

course.courseid-----score.courseid

3.SQL

- WHAT(SQL是什么?)
 - Structured Query Language: 结构化查询语言
- WHY(为什么要使用SQL?)
 - 难道仅仅使用SQL Server Management Studio操作数据库?
 - 应用程序如何与数据库打交道?
- WHEN(何时使用?)
 - 对SQL Server执行所有的操作都可以
 - 程序中的增删改查
- HOW(怎么使用?)
 - ...

3.1 四大SQL语句

四大SQL语句

- --1. 数据定义语句DDL:create、 alter、 drop 、 truncate（表结构）
- --2. 数据操作语句DML: insert、 delete、 update、 select（数据）
- --3. 数据控制语句DCL: 授权grant 收回权限: revoke
- --4. 事务控制语句TCL:
 - 开启事务: begin
 - 提交: commit ,
 - 回滚: rollback



4. 1 创建数据库

SQL语句：

```
create database 数据库名;
```

```
例： create database stu;
```

成功创建数据库后，数据库根目录下会自动创建数据库目录。



4. 2 显示数据库结构

SQL语句：

```
show create database 数据库名;
```

可以查看数据库的相关信息（例如MySQL版本ID号、默认字符集等信息）。

```
mysql> show create database student;
+-----+-----+
| Database | Create Database
+-----+-----+
| student | CREATE DATABASE `student` /*!40100 DEFAULT CHARACTER SET latin1 */
+-----+-----+
1 row in set (0.00 sec)
```

4. 3 删除数据库

SQL语句：

drop database 数据库名;

```
mysql> drop database student;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> show databases;
```

Database
information_schema
choose
mysql
performance_schema
test

```
5 rows in set (0.00 sec)
```



5. 1 MySQL 数据类型

MySQL提供的数据类型包括数值类型（整数类型和小数类型）、字符串类型、日期类型、复合类型（复合类型包括enum类型和set类型）以及二进制类型。

整型	<code>tinyint</code>	1个字节
	<code>smallint</code>	2个字节
	<code>mediumint</code>	3个字节
	<code>int</code>	4个字节
	<code>bigint</code>	8个字节
小数类型	<code>decimal</code>	
	<code>float</code>	4个字节
	<code>double</code>	8个字节
字符类型	<code>char</code>	定长字符串
	<code>varchar</code>	变长字符串
	<code>text</code>	<code>tinytext, text, mediumtext, longtext</code>
日期类型	<code>date</code>	默认格式: <code>YYYY-MM-DD</code>
	<code>time</code>	<code>HH:ii:ss</code>
	<code>year</code>	
	<code>datetime</code>	<code>YYYY-MM-DD HH:ii:ss</code>
	<code>timestamp</code>	日期与时间的混合数据类型

类型	字节数	范围（有符号）	范围（无符号）
tinyint	1字节	(-128, 127)	(0, 255)
smallint	2字节	(-32768, 32767)	(0, 65535)
mediumint	3字节	(-8388608, 8388607)	(0, 16777215)
int	4字节	(-2147483648, 2147483647)	(0, 4294967295)
bigint	8字节	(-9233372036854775808, 9223372036854775807)	(0, 18446744073709551615)

类型	字节数	负数的取值范围	非负数的取值范围
float	4	-3.402823466E+38到-1.175494351E-38	0和1.175494351E-38到3.402823466E+38
double	8	-1.7976931348623157E+308到- 2.2250738585072014E-308	0和2.2250738585072014E-308到 1.7976931348623157E+308



decimal(length, precision)用于表示精度确定（小数点后数字的位数确定）的小数类型，length决定了该小数的最大位数，precision用于设置精度（小数点后数字的位数）。

例如：

decimal (5,2) 表示小数取值范围：999.99 ~ 999.99

decimal (5,0) 表示： -99999 ~ 99999 的整数。



char()与varchar():

例如对于简体中文字符集gbk的字符串而言，`varchar(255)`表示可以存储255个汉字，而每个汉字占用两个字节的存储空间。假如这个字符串没有那么多汉字，例如仅仅包含一个‘中’字，那么`varchar(255)`仅仅占用1个字符（两个字节）的储存空间；而`char(255)`则必须占用255个字符长度的存储空间，哪怕里面只存储一个汉字。



6.1 创建数据库表

注意：在创建表之前，需要选择当前操作的数据库；

Use 数据库名；

创建数据库表SQL语句：

Create table 表名 (字段名1, 数据类型 [约束条件],

...

[其他约束条件]

....) 其他选项 (例如存储引擎、字符集等选项)

；

例如：

```
use student;  
create table stu( id int,  
    name varchar(20)  
);
```



6. 2 定义约束 (Constraint) 条件

常见的约束条件有6种

主键 (Primary Key) 约束

外键 (Foreign Key) 约束

唯一性 (Unique) 约束

默认值 (Default) 约束

非空 (Not NULL) 约束

检查 (Check) 约束

MySQL不支持check约束，写了没有作用，但不会报错

添加约束的方法：

1、可以在创建表的时候添加；

2、可以用alter语句添加；

例： alter table stu add constraint un_id unique(id);

alter table stu add unique(id);

举例：

```
create table stuinfo(
    id int not null auto_increment,##设置自增， 非空值
    name char(20) not null,
    address char(50),
    city char(50),
    age int not null,
    love char(50) not null default 'No habbit',##设置默认值
    primary key(id)###设置id为主键
)engine=InnoDB DEFAULT CHARSET = utf8;
```



6. 3 显示表结构

SQL语句：

desc 表名;-----即可查看指定表的结构

SQL语句：

show create table 表名;-----查看指定表的详细信息。

6. 4 删除表

SQL语句：

```
drop table 表名;
```

注意：删除表后，MySQL服务实例会自动删除该表结构定义文件，以及数据、索引信息。该命令慎用！



6. 5 修改字段相关信息

1. 删除字段

删除表字段的语法格式如下。

```
alter table 表名 drop 字段名
```

2. 添加新字段

```
alter table 表名 add 新字段名 新数据类型 [ 新约束条件 ]  
[ first | (after 旧字段名) ]
```

3. 修改字段名

```
alter table 表名 change 旧字段名 新字段名 新数据类型
```

4、或者数据类型

```
alter table 表名 modify 字段名 新数据类型
```

```
mysql> alter table stu drop name;
```

```
mysql> alter table stu add name varchar(20);
```

```
mysql> alter table stu change name newname varchar(20);
```

```
mysql> alter table stu modify newname varchar(30);
```



6. 6修改表名

SQL语句

```
rename table 旧表名 to 新表名
```

该命令等效于： alter table 旧表名 rename 新表名

7.1 表记录的插入

SQL语句：

```
insert into 表名 [ (字段列表) ] values (值列表)
```

提示：当插入的数据值的个数与表字段个数相同时，可以省略字段列表；

```
Insert stu(id,name) values(1,'newdream');
```

```
Insert stu values(2,'newdream1');
```



7.2 表记录的插入（一次插入多条）

SQL语法：

```
insert into 表名[(字段列表)] values  
    (值列表1),  
    (值列表2),  
    ...  
    (值列表n);
```

```
Insert stu values(3,'new1'), (4,'new2'), (5,'new3');
```

7.3 使用insert...select插入结果

SQL语句：

```
insert into 目标表名[(字段列表1)]
select (字段列表2) from 源表 where 条件表达式
```

```
例： insert into stu_info(id,name)
select (id,newname) from stu where id>2;
```

注意：字段列表1与字段列表2的字段个数必须相同，且对应字段的数据类型尽量保持一致。
如果源表与目标表的表结构完全相同，“(字段列表1)”可以省略。

7.4 使用replace插入新记录

replace语句的语法格式有三种语法格式。

replace into 表名 [(字段列表)] values (值列表)

replace [into] 目标表名[(字段列表1)]
select (字段列表2) from 源表 where 条件表达式

replace [into] 表名 set 字段1=值1, 字段2=值2



7.5 复制表结构和内容到另一张表中

1、复制表结构及数据到新表

```
Create table 新表 select * from 旧表;
```

2、只复制表结构到新表

```
Create table 新表 select * from 旧表 where 1=2;
```

8. 表记录的修改

SQL语句：

```
update 表名 set 字段名1=值1,字段名2=值2,...,字段名n=值  
n [where 条件表达式]
```

```
update stu set newname='new5' where id=1;
```

where子句指定了表中的哪些记录需要修改。若省略了where子句，则表示修改表中的所有记录。

set子句指定了要修改的字段以及该字段修改后的值。

9.1 表记录的删除

1、使用**delete**删除表记录

SQL语句：

```
delete from 表名 [where 条件表达式]
```

说明：如果没有指定**where**子句，那么该表的所有记录都将被删除，但表结构依然存在。

2、使用**truncate**清空表记录

SQL语句：

```
truncate [table] 表名
```

9.2 delete和truncate的区别

Delete不加WHERE条件是删除所有数据

Truncate不能够加WHERE条件

Delete可以加WHERE条件

Truncate会重置AUTO_INCREMENT

Delete可以进行回滚操作

10. 表记录的查询

SQL语法：

```
select 字段列表 from 表名 [ where 条件表达式 ]
```

字段列表	说明
*	字段列表为数据源的全部字段。
表名.*	多表查询时，指定某个表的全部字段。
字段列表	指定所需要显示的列。



表记录的查询—给列取别名

可以为字段列表中的字段名或表达式指定别名，中间使用as关键字分隔即可（as关键字可以省略）。
多表查询时，同名字段前必须添加表名前缀，中间使用“.”分隔。

```
Select id as '学生学号', newname '学生姓名' from stu;
```

总结

- 1、了解MySQL的两种存储引擎；
- 2、熟练掌握基本的SQL语句，如建表建库，对表中的数据进行增删改查；