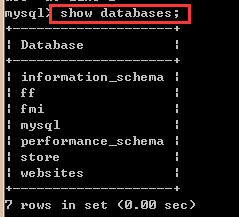
## 1 操作数据库

### 1 登录数据库

* mysql -uroot -p
* mysql -h 127.0.0.1 -uroot -p123 #使用帐号root,密码为123,登录ip地址为127.0.0.1的主机的数据库

### 2 查看数据库文件

* show databases;



### 3 创建数据库

创建名为 test 的数据库

* create database test;

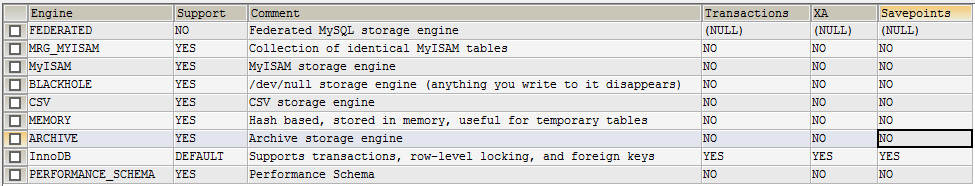
### 4 删除数据库

删除名为 test 的数据库

* drop database test;

### 5 查看数据库引擎

* show engines;



查询结果中，

* + - Engine 参数指存储引擎名称；
    - Support 参数说明MySQL是否支持该类型引擎；
    - Comment 参数表示对该引擎的评论；
    - Transaction 参数表示是否支持事务处理；
    - XA 参数表示是否分布式交易处理的XA规范；
    - Savepoints 参数表示是否支持保存点，以方便事务的回滚操作；

由上面我们看到InnoDB存储引擎是default的，也就是数据库默认的存储引擎。

InnoDB是MySQL的一种存储引擎，InnoDB给MySQL提供了事务、回滚、崩溃修复能力和多版本并发控制的事务安全。InnoDB是MySQL上第一个提供外键约束的表引擎，而且对事务处理的能力，也是其他存储引擎不能比拟的。不过这种引擎的缺点就是读写效率稍差，占用的数据空间相对比较大。

## 2 表操作

### 1 创建表

CREATE TABLE 表名 (

属性名 数据类型 [完整约束条件],

属性名 数据类型 [完整约束条件],

...

...

属性名 数据类型 [完整约束条件]

);

示例:

CREATE TABLE fengmi (\_id INT PRIMARY KEY AUTO\_INCREMENT,

\_name VARCHAR(20),

age INT DEFAULT 0 );



上面创建表的时候涉及到一个完整性约束条件，下面就列出一个完整性约束条件表

|  |  |
| --- | --- |
| 约束条件 | 说明 |
| PRIMARY KEY | 标识该属性为该表的主键，可以唯一的标识对应的元组 |
| FOREIGN KEY | 标识该属性为该表的外键，是与之联系某表的主键 |
| NOT NULL | 标识该属性不能为空 |
| UNIQUE | 标识该属性的值是唯一的 |
| AUTO\_INCREMENT | 标识该属性的值是自动增加，这是MySQL的SQL语句的特色 |
| DEFAULT | 为该属性设置默认值 |

### 2 设置表的外键

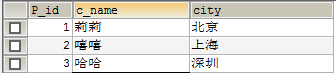
* 创建表时,直接指定外键

1 创建persons表

CREATE TABLE persons (P\_id INT PRIMARY KEY AUTO\_INCREMENT,

c\_name VARCHAR(20),

city VARCHAR(20));



2 创建表orders

CREATE TABLE orders (

O\_id INT NOT NULL AUTO\_INCREMENT,

order\_no VARCHAR(20),

P\_id INT DEFAULT NULL,

PRIMARY KEY (O\_id), # 指定主键

CONSTRAINT fk\_persons FOREIGN KEY (P\_id) REFERENCES persons(P\_id) #指定外键约束

)



请注意，"orders" 表中的 "P\_id" 列指向 "persons" 表中的 "P\_Id" 列。

"persons" 表中的 "P\_Id" 列是 "persons" 表中的 PRIMARY KEY(主键)。

"orders" 表中的 "P\_Id" 列是 "orders" 表中的 FOREIGN KEY(外键)。

FOREIGN KEY 约束用于预防破坏表之间连接的行为。

FOREIGN KEY 约束也能防止非法数据插入外键列，因为它必须是它指向的那个表中的值之一,

例如向表orders中执行下面插入语句时,将报错

INSERT INTO orders VALUES(NULL,'667788',5);

Error Code : 1452

Cannot add or update a child row: a foreign key constraint fails (`foreign\_key\_demo`.`orders`, CONSTRAINT `fk\_persons` FOREIGN KEY (`P\_id`) REFERENCES `persons` (`P\_id`))

因为在表persons中的P\_id列不存在值为5的行,所以插入orders中外键列P\_id的值不能为5!!!

* 当 "Orders" 表已被创建时，如需在 "P\_Id" 列创建 FOREIGN KEY 约束，请使用下面的 SQL

ALTER TABLE orders

ADD FOREIGN KEY (P\_Id)

REFERENCES persons(P\_Id)

### 3 撤销外键约束

1 查看要撤销外键约束的别名

SHOW CREATE TABLE orders; # 查看表orders的创建信息

数据库输出信息如下:

| orders | CREATE TABLE `orders` (

`O\_id` int(11) NOT NULL AUTO\_INCREMENT,

`order\_no` varchar(20) DEFAULT NULL,

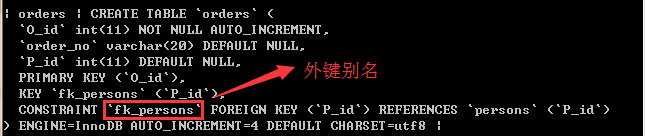
`P\_id` int(11) DEFAULT NULL,

PRIMARY KEY (`O\_id`),

KEY `fk\_persons` (`P\_id`),

CONSTRAINT `fk\_persons` FOREIGN KEY (`P\_id`) REFERENCES `persons` (`P\_id`)

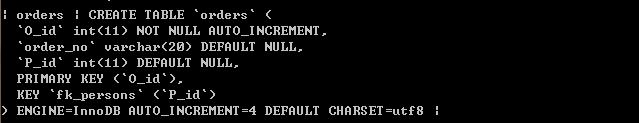
) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8 |



2 通过外键别名移除对应的外键约束

ALTER TABLE orders **DROP** FOREIGN KEY fk\_persons; #撤销别名为fk\_persons的外键约束

再次查看表的创建信息,可以看到外键约束已经不存在.



### 4 删除表

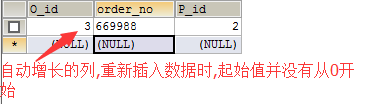
语法: DROP TABLE 表名;

DROP TABLE orders; # 将表orders从数据库中删除

### 5 删除表数据

* DELETE FROM orders; # 将表order中的所有数据删除 ,

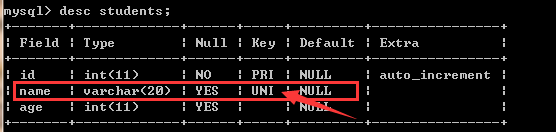
这种方式数据会被全部删除,但是设置 auto\_increment的列,其起始值并没有被重置,即下次插入的数值将从上次记录的值基础上加1



* **TRUNCATE** TABLE orders; # 删除orders表中的所有数据,相当于删表后重建表

### 6 撤销 UNIQUE 约束

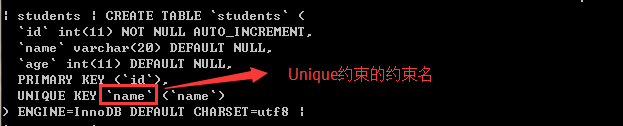
有如下表



可以看到,表中的列 name 设置了 唯一约束 UNIQUE,撤销方式如下:

1 通过查看表的创建信息,获取唯一约束的**约束名**

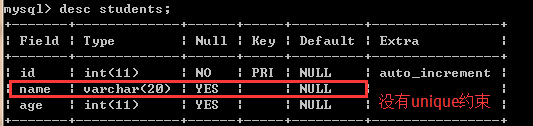
SHOW **CREATE** TABLE students; # 查看表的创建信息



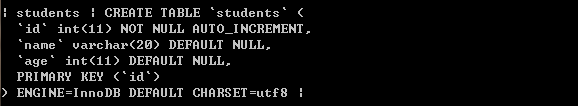
2 通过约束名撤销约束

ALTER TABLE students **DROP INDEX** name; # 撤销约束名为 name 的约束

撤销后,再次查看表信息:



表的创建信息:



### 7 修改auto\_increment的起始值

ALTER TABLE persons AUTO\_INCREMENT=100 # 将表persons自动增长列的起始值设为100

## 3 sql聚合函数

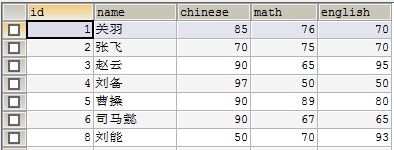
SQL Aggregate 函数计算**从列中取得**的值，**返回**一个单一的**值**

**处理对象:列数据**

**返回:单一的值**

* AVG() - 返回平均值
* COUNT() - 返回行数
* FIRST() - 返回第一个记录的值
* LAST() - 返回最后一个记录的值
* MAX() - 返回最大值
* MIN() - 返回最小值
* SUM() - 返回总和

表数据如下:



### 1 AVG() 取平均值

1 输出每一学科的平均分

SELECT **AVG**(chinese) AS '语文平均分', #取chinese列的平均值,使用别名 ‘语文平均分’显示

**AVG**(math) AS '数学平均分', #取math列的平均值

**AVG**(english) AS '英语平均分' #取english列的平均值,使用别名显示

FROM exam;

输出结果:



2 输出chinese分数大于平均分数的信息

思路: select ‘需要的信息的列名’ from 表明 where ‘分数大于语文平均分’

SELECT id,NAME,chinese FROM exam

WHERE (chinese > (SELECT **AVG**(chinese) FROM exam));

输出结果:



### 2 COUNT() - 返回行数

1 返回表中记录的行数

SELECT **COUNT**(\*) FROM exam; #查询表exam中记录的行数 ---> 输出结果 7

2 返回表中english的行数(含重复值的情况)

SELECT **COUNT**(english) FROM exam; ---> 输出结果 7

3 返回表中english的行数(去重)

SELECT **COUNT**(**DISTINCT** english) FROM exam; ---> 输出结果 6

### 3 FIRST() - 第一个记录的值

FIRST() 函数返回指定的列中第一个记录的值, MySQL数据库不支持,MySQL可以使用Limit 关键字实现

1 返回表exam中chinese分数最高的人的信息

* 思路1:先按chinese列数据降序排列,然后取第一行数据

SELECT \* FROM exam

ORDER BY chinese DESC # 按chinese降序排列

**LIMIT** 1; # 选第一行数据

* 思路2:选取chinese列的值等于最大值的那一行

SELECT \* FROM exam

WHERE chinese=(SELECT **MAX**(chinese) FROM exam);

2 分页查询

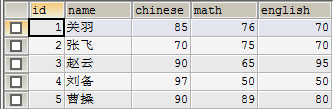
通过limit 可以限制单次查询返回数据的条数

语法: limit total --> 返回 total条数据

* + 从exam表中查询前5条数据

SELECT \* FROM exam LIMIT 5; # 返回前5条数据

输出结果:

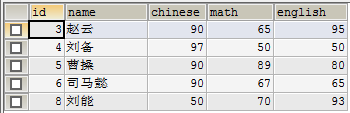


* + 带偏移量的数据条数

语法: LIMIT startIndex total --> 从 startIndex算起total条数据

SELECT \* FROM exam LIMIT 2,5; # 返回数据库中第二条数据后面的5条数据

输出结果:



### 4 SUM - 返回总和

SUM() 函数返回数值列的总数

1 返回math列的总和

SELECT **SUM**(math) AS ‘列math的总和’ FROM exam;

输出结果:



### 5 MAX() - 返回最大值

MAX() 函数返回指定列的最大值。

语法:

SELECT **MAX**(column\_name) FROM table\_name;

### 6 MIN() - 返回最小值

MIN() 函数返回指定列的最小值.

语法:

SELECT **MIN**(column\_name) FROM table\_name;

### 7 GROUP BY - 按..分组

GROUP BY 语句常用于结合聚合函数，根据一个或多个列对结果集进行分组

语法:

SELECT 列名, 聚合函数(列名)

FROM 表名

WHERE 列名 operator value

GROUP BY 列名;

**表t**数据如下:



1 统计每个人的考试科目总数

SELECT name,**COUNT**(course) AS 考试科目总数

FROM t

**GROUP BY** name;

输出结果:



### 8 HAVING

在 SQL 中增加 HAVING 子句原因是，WHERE 关键字无法与聚合函数一起使用。

HAVING 子句可以让我们筛选分组后的各组数据.

语法:

SELECT column\_name, 聚合函数(column\_name)

FROM table\_name

WHERE column\_name operator VALUE

GROUP BY column\_name

**HAVING** 聚合函数(column\_name) operator VALUE;

1 查询表t中考了3门试的学生姓名

SELECT name FROM t

GROUP BY name # 按姓名分组

**HAVING** COUNT(course)>2; # 科目总数大于2

2 查询出每门课都大于80分的学生的姓名

SELECT name FROM t

GROUP BY name # 按姓名分组

HAVING MIN(score)>80; # 分数不低于80